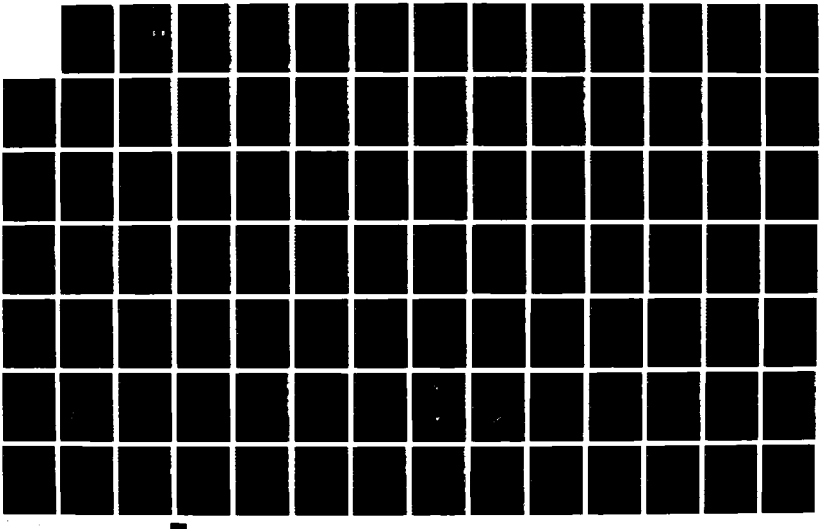
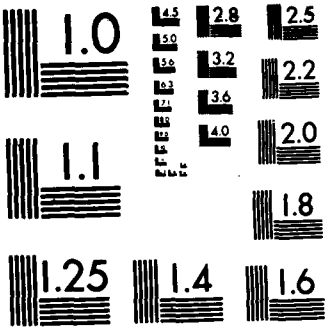
AD-A193 377 OPTIMUM-PATH-TO-GO GUIDANCE OF COMMAND ADJUSTED TRAJECTORY PROJECTILES(U) BARRON ASSOCIATES INC STANARDSVILLE VA R L BARRON ET AL. 22 JAN 88 129-F 1/2

UNCLASSIFIED DAAA21-87-C-0140 F/G 17/7 NL





Final Technical Report
Contract DAAA21-87-C-0140

OPTIMUM-PATH-TO-GO GUIDANCE OF COMMAND ADJUSTED TRAJECTORY PROJECTILES

Roger L. Barron
John F. Elder IV

January 22, 1988

Submitted to:

Commander
U.S. Army ARDEC
Attention: SMCAR-CCS-A (Building 1)
Picatinny Arsenal, NJ 07806-5001

Submitted by:

BARRON ASSOCIATES, INC.
Route 1, Box 159
Stanardsville, Virginia 22973
(804) 985-4400

"The view, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation."

Final Technical Report
Contract DAAA21-87-C-0140

OPTIMUM-PATH-TO-GO GUIDANCE OF COMMAND ADJUSTED TRAJECTORY PROJECTILES

Roger L. Barron
John F. Elder IV

January 22, 1988

Submitted to:

Commander
U.S. Army ARDEC
Attention: SMCAR-CCS-A (Building 1)
Picatinny Arsenal, NJ 07806-5001

Submitted by:

BARRON ASSOCIATES, INC.
Route 1, Box 159
Stanardsville, Virginia 22973
(804) 985-4400

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) 129-F	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Barron Associates, Inc. 6b. OFFICE SYMBOL (If applicable) OATL7	7a. NAME OF MONITORING ORGANIZATION U. S. Army Armament, Munitions and Chemical Command
6c. ADDRESS (City, State, and ZIP Code) Route One, Box 159 Stanardsville, Virginia 22973-9511	7b. ADDRESS (City, State, and ZIP Code) Department of the Army AMCCOM Picatinny Arsenal, New Jersey 07806-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION same as 7a 8b. OFFICE SYMBOL (If applicable) SMCAR-CCS-A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAA21-87-C-0140
8c. ADDRESS (City, State, and ZIP Code) same as 7b	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
Optimum-Path-To-Go Guidance of Command Adjusted Trajectory Projectiles

12. PERSONAL AUTHOR(S)
Roger L. Barron and John F. Elder IV

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final Technical	FROM 87/7/29 TO 88/1/28	January 22, 1988	115

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	maneuverable projectiles; two-point boundary-value problem; tactical weapon systems; microprocessor applications; polynomial networks; optimum control; optimum guidance

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Execution of optimal guidance and control algorithms in real time with a microprocesssor is feasible using polynomial networks to store, in compact and almost instantly retrievable form, information about a large number of pre-computed optimum two-point boundary-value (TPBV) trajectory solutions. The mathematical basis for computing these optimum solutions has been derived in this SBIR Phase I work using the calculus of variations to obtain adjoint differential equations that are initialized for the TPBV solution. Historically, these initializations have usually necessitated iterative computations because the adjoint equations are not analytically integrable. It is shown that polynomial networks fitted, off-line, to a data base of optimum trajectories can provide the appropriate initializations in real time. These polynomial networks offer optimum-path-to-go (OPTG) and target motion prediction capabilities that can significantly improve performance of maneuverable projectiles, particularly in cases of thruster impulse

continued on reverse side

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Grant W. Manning, Jr.	22b. TELEPHONE (Include Area Code) 201 724-4778 22c. OFFICE SYMBOL SMCAR-CCS-A

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

19. ABSTRACT (continued)

limitations and substantial uncertainties about target behavior. This report presents the derivation of the governing variational equations, a simulation for three-degree-of-freedom trajectory guidance analyses of command adjusted trajectory (CAT) projectiles engaging maneuvering targets, and results for a 120mm CAT projectile using (1) a predictive proportional navigation (baseline) guidance law, (2) OPTG guidance initialization at t = 0 via a polynomial network (using no subsequent re-initialization), and (3) blended commands using OPTG guidance initialized at t = 0 (only) until the estimated time-to-go is less than 1.0 sec., then switching to the baseline law for terminal guidance. The results show that the blended guidance law has a smaller miss distance and greater divert capability than the baseline guidance, and that pure OPTG guidance requires fewer divert squib firings than the baseline guidance. It is estimated that implementation of the pure OPTG or blended guidance requires a processor throughput of approximately 5.3 kFLOPS exclusive of sensor data pre-processing, target tracking and prediction computations, and computing overhead.

FOREWORD

This is the final technical report submitted by Barron Associates, Inc., Stanardsville, Virginia, under Phase I SBIR Contract DAA21-87-C-0140 ("Use of Polynomial Networks to Improve on Control Efficiency of Maneuverable Projectile") with the U. S. Army Armament, Munitions and Chemical Command, Picatinny Arsenal, New Jersey. The work reported was performed during the period July 29 - December 28, 1987. The Principal Investigator was Roger L. Barron, who is responsible for the optimum-path-to-go guidance concept. The programming and numerical analyses were performed primarily by John F. Elder IV. Dean W. Abbott assisted in the work.

The Contracting Officer's Representatives in this project were Thomas D. Hutchings SMCAR-CCS-C (July 29 - October 28) and Grant W. Manning, Jr., SMCAR-CCS-A (October 29 - completion). The authors are indebted to them and to Modesto J. Barbarisi, SMCAR-SCF-RE, Cliff Wilkins, and Al Rahe for technical data, encouragement, guidance, and support.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

LIST OF FIGURES

1. SUMMARY AND INTRODUCTION

1.1. Summary

An optimum-path-to-go (OPTG) guidance capability has been designed in preliminary form and investigated to demonstrate its feasibility for command adjusted trajectory (CAT) projectiles. The principal design tools used are the calculus of variations and an advanced algorithm for synthesis of polynomial networks from simulation data. Differential equations governing optimum two-point boundary-value projectile guidance solutions for maneuvering targets are derived via the calculus and initialized in real-time calculations using an adaptively-synthesized polynomial network (APN). Application of OPTG guidance to 120mm projectiles has been simulated. In particular, comparisons have been made of the performance of the OPTG guidance when it is blended with a predictive proportional navigation (baseline) guidance law. In the blended law, pure OPTG guidance is initialized at t = 0 and used until estimated time-to-go is less than or equal to 1.0 sec. (No re-initialization of the OPTG guidance is used after t = 0.) The baseline guidance is then employed after time-to-go becomes less than 1.0 sec. until impact (or closest point of approach).

Block diagrams of the OPTG and baseline guidance laws are presented below in Figures 1 and 2. Figure 1.a illustrates an arrangement in which the means for intercept point prediction are separate from those for computing the initialization (and re-initialization) solutions, which is the case investigated in the present study. Figure 1.b shows the means for intercept point prediction combined with those for the initialization/re-initialization computations, which is the configuration recommended for future development. Figure 2 (in Section 5) presents a block diagram of the baseline guidance law.

Simulations of the blended guidance law and pure OPTG guidance have demonstrated the following performance benefits for target trajectories having a small level of uncertainty concerning the target accelerations and intercept coordinates:

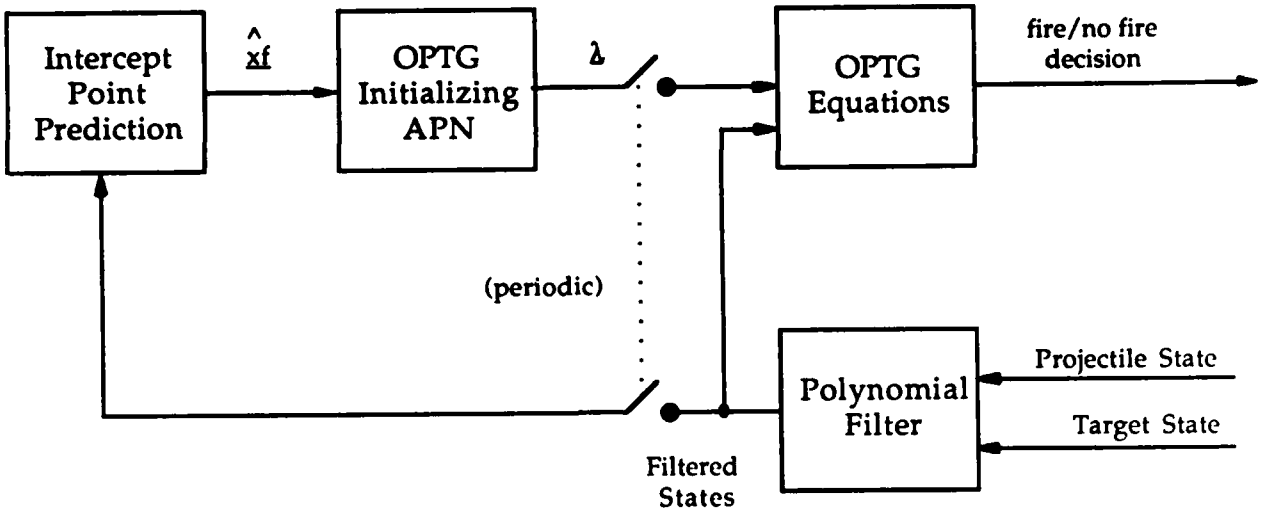


Figure 1.a: OPTG Guidance Computations

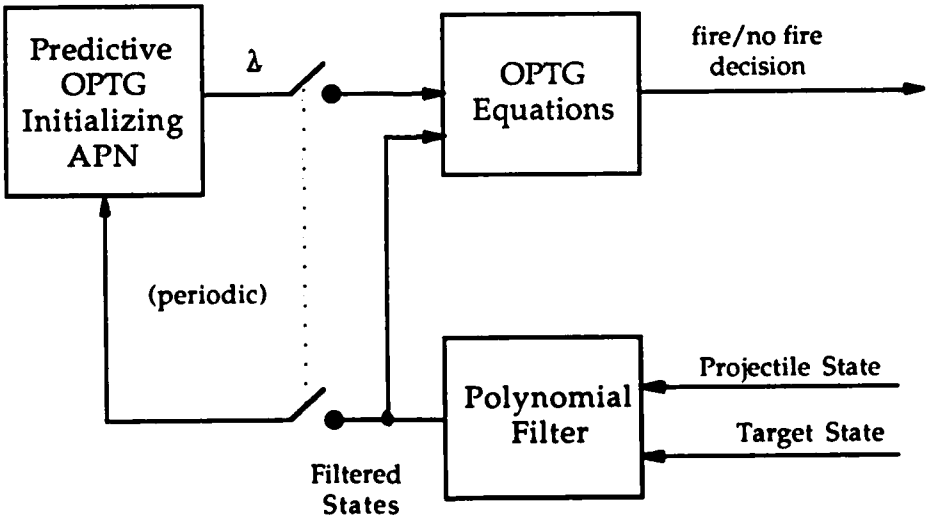


Figure 1.b: Condensed OPTG Guidance Computations

1. The average closest-point-of-approach (CPA) is 2.0 times less for the blended guidance law than for the baseline.

2. The area in the miss-distance plane reachable with a CPA less than 6 ft. is 3.5 times greater for the blended guidance law than for the baseline.

3. For all values of required divert, the average improvement ratio, defined as CPA for the baseline divided by CPA for the blended guidance law, is greater than 1.0. For required divert distances between 22 and 54 ft., the ratio is greater than 2.0; and between 30 and 51 ft., the ratio is greater than 5.0.

4. For the pure OTG guidance, a given divert is achieved with 10 fewer squib firings on the average than are used with the baseline guidance. When the blended guidance law is employed, squib utilization is equal for the blended guidance law and the baseline. (It is believed that re-initialization of the OPTG guidance after t = 0 in the blended guidance law will produce most of the squib-usage reduction of pure OPTG guidance.)

1.2 Background

When using conventional guidance (such as the baseline guidance law), a critical consideration is the establishment of the threshold used for the squib fire/no-fire decisions. If this threshold is set too low, the system may use an excessive number of squibs when the total divert required is small or even negligible. To illustrate this point, a simulation of the baseline guidance has been performed for a case in which no divert was required; that is, the gun was perfectly aimed and purely ballistic motion of the projectile would score an exact hit against a non-moving airborne target (such as a helicopter in stationary hover). With the decision threshold of the baseline law set to zero, the projectile used almost all of its 40 squibs. (It became, in effect, a self-disturbing as well as self-correcting system, a system triggered by the slightest noise.) Although the miss distance was nearly zero, the zero-level threshold

clearly produced a design that was profligate in its use of maneuver resources -- a design that could be quite readily defeated by target maneuvering.

On the other hand, if the decision threshold is set too high, the baseline guidance fails to react as early in the engagement as may be necessary if a large divert is required. As a further illustration, the threshold of the baseline guidance was next set to a moderate positive level so as to reduce unnecessary squib firings for the case just described by about 50 percent. Miss distance remained small as long as little or no divert was required, but the design exhibited increased miss distances when stressed; that is, required to approach its maximum total divert capabilities. The conclusion: threshold selection for conventional guidance is at best a compromise between conditions of small and large required divert. It can be shown also that this threshold selection is a compromise between conditions of small and large uncertainties (in measurements of system states and predictions of target motions).

An important attribute of OPTG guidance is that the selection of its decision threshold is independent of the amount of divert that will be required. To demonstrate this, the OPTG guidance was simulated for the two extreme cases mentioned above (no divert required and maximum divert required). The threshold was kept at zero for the OPTG tests. When no divert was required, no squibs were fired. When maximum divert was required, the OPTG guidance began squib firing at the earliest available opportunity (as afforded by the cyclical variation of roll attitude), and kept firing at each succeeding opportunity until the supply of squibs was exhausted. Clearly, no strategy for guidance could have produced a greater divert. Between the limits of zero and maximum divert, the OPTG guidance used a number of squibs that was approximately in proportion to the required divert. At no time was there any question regarding the choice of decision threshold.

For the OPTG guidance system, the decision threshold can be set entirely on the basis of the expected levels of noise and target maneuver uncertainty. If the threshold is placed above zero in consideration of these expected levels, there will be some loss of maximum divert capability. But the OPTG guidance is appropriately nonlinear, and small decision thresholds

incorporated to de-sensitize it for realistic levels of noise and uncertainty have an almost negligible influence on the maximum divert capability. Conversely, the baseline guidance law is proportional in its action, and the same threshold that adequately de-sensitizes both it and the OPTG guidance to the effects of noise and target motion uncertainties can be crippling to the divert capability of the baseline guidance.

1.3 Optimum-Path-To-Go-Guidance

Optimum-path-to-go guidance provides optimum, real-time, two-point boundary-value guidance. It periodically computes a new best trajectory from the actual position and velocity state of the weapon to a designated final position. In application to CAT projectiles, the OPTG method offers two particularly important benefits. First, by maneuvering throughout the engagement to establish the designated final state, the magnitude of required terminal acceleration is minimized, greatly reducing miss distance. Second, the OPTG guidance realizes optimum performance between specified boundary conditions and thus enlarges the achievable boundaries in comparison with traditional guidance.

OPTG guidance uses polynomial networks, evolved from neural networks beginning in the early 1960s [1-3, 5-8, 12, 16], to implement, via algebraic inverse modeling, time-varying solutions for the weapon steering commands as functions of initial, i.e., "time-now", and designated future trajectory states. (In trajectory prediction, information about initial states and the steering commands is used to compute a future state or states. In inverse modeling, the boundary conditions are supplied and the commands that satisfy those conditions are computed.) As do other neural networks, the polynomial network learns from experience. However, the learning process is completed off-line during the design phase. The network that is used in the weapon thus has fixed parameters, and its behavior is fully known (primarily from simulations but also from flight testing) before operational usage occurs.

The calculus of variations is employed in design of the guidance law so as to achieve optimality and the benefits of a compact representation of the

optimum trajectories. The procedure is first to derive the governing variational equations, which comprise, chiefly, a family of first-order, nonlinear, time-varying *adjoint* differential equations involving the Lagrange multipliers of the formulation. The derivation is based upon an integral performance criterion having a variable upper limit (final time) of integration. Next, an extensive data base of optimum trajectories is computed. Each entry in this data base consists of an initial condition (i.e., the initial position and velocity states), a set of initializing Lagrange multiplier values (which determine the specific optimal trajectory flown as the adjoint equations are integrated), and the resulting final condition (final states). The data base may be thought of as a field of extremals between large envelopes of admissible initial and final condition pairs. The Algorithm for Synthesis of Polynomial Networks (ASPN) is then used to create, off-line, a nonlinear polynomial network transformation (called an adaptively-synthesized polynomial network or APN) that maps observed initial and desired final states into initializing multiplier values that produce an optimum path between the prescribed states. Within the weapon, the APN may be interrogated periodically to obtain optimum-path-to-go solutions, and the adjoint equations are integrated to compute steering commands until the next re-initialization is performed.

Potential further uses of polynomial networks are to (1) predict the target trajectory and (2) estimate sensitivities of weapon predicted final states to changes in initial values of the Lagrange multipliers, the latter providing the means for closed-loop vernier adjustments of guidance commands so as to null the predicted final error.

ASPN [12] evolves the network during the off-line design, beginning with the simplest structure (each output connected directly to the best corresponding input) and growing to an appropriately complex structure. The final network typically involves several layers of nodal algebraic elements, each having one, two, or three inputs obtained from a previous layer and providing a nonlinear analytic function that includes cross-products of its inputs. An information theoretic modeling criterion is used by ASPN during the synthesis process to select the most relevant inputs and combine them in suitable elements and in a suitable overall network transformation

that is learned during synthesis. The criterion also regulates the allowed complexity of the network to avoid overfitting the data (over-specialization). The ASPN procedure has been undergoing development for 27 years and has been extensively proven in signal processing, prediction, and control applications.

1.4 Work Accomplished

The specific accomplishments of this Phase I work are:

1. The three-degree-of-freedom (3DOF) translational equations of motion for a 120 mm maneuverable CAT projectile have been written and programmed in a simulation for trajectory data base generation. This simulation includes provisions for target maneuvers and maneuver uncertainties.

2. A baseline predictive guidance law has been formulated and used for comparative analysis purposes.

3. The governing equations for optimum-path-to-go (OPTG) guidance of the CAT projectile have been derived using an extended calculus of variations methodology. Performance sub-criteria in this derivation of the two-point boundary-value guidance solution include (a) minimization of thruster firings, (b) maximization of projectile range, (c) maximization of projectile divert capability, (d) conservation of projectile kinetic energy, and (e) trajectory shaping factors associated with the levels of uncertainty regarding system states and future maneuvers of the target.

4. The OPTG guidance equations have been incorporated in the 3DOF simulation and numerical studies performed, producing the resuilts summarized in Section 1.1 above.

5. Processor throughput required to implement the OPTG guidance has been estimated. (The throughput requirement is approximately 5.3 kFLOPS.)

2. CRITERION OF PERFORMANCE

The Lagrange formulation of the performance criterion involves, in this application, the minimization of the definite integral

$$I = \int_{t_i}^{t_f} F\,dt \qquad 2:1$$

where t_i is the time of guidance computation (at guidance initiation $t_i = t_0$) and t_f is the final (burst or impact) time. From the mathematical viewpoint, time t_i is fixed, but t_f is variable and subject to optimization. The integrand (Hamiltonian) may be written as

$$F = \sum_{n=0}^{n'} W_n G_n + \sum_{j=1}^{j'} \lambda_j f_j \qquad 2:2$$

in which the W_n's are constant Lagrange multipliers that serve as criterion weights; the G_n's are the integrands of isoperimetric conditions that comprise sub-criteria performance measures; and the λ_j's are, in general, time-varying Lagrange multipliers that introduce the anholonomic (non-integrable) equations of motion and kinematic relationships.

It is assumed that F(t) is continuous and has continuous second partial derivatives with respect to the state variables of the physical system to be guided.

The following sub-criteria are proposed.

2.1 Guidance Effort Penalty

Let N(t) represent the cumulative number of squib firings by the projectile as of time t. Then, if no squib firing occurs at t, $\dot{N}(t)$ is zero, and if a squib is fired, $\dot{N}(t)$ is unity over the short finite interval (t, t + Δt) between t

and the next decision point. Note that N(t) is continuous even though $\dot{N}(t)$ can exhibit jump discontinuities between the levels zero and one.

The total guidance effort over the flight of the projectile is defined as

$$\int_{t_0}^{t_f} \dot{N}^2 \, dt$$

In the context of optimum-path-to-go guidance, the guidance effort to be expended over the time remaining as of a solution time t_i is

$$\int_{t_i}^{t_f} \dot{N}^2 \, dt$$

This integral is seen to be independent of when squibs are fired and to vary only with the number of firings over the interval (t_i, t_f). It may be advantageous to fire the squibs early in the engagement, so as to obtain maximum divert capability for the expenditure of resources. Conversely, it may be advantageous to fire the squibs late in the engagement, so as to use resources when there is the least uncertainty about where the target will be at t_f. One way to achieve a specified time weighting of the guidance effort is to take

$$\int_{t_i}^{t_f} \tau^r \dot{N}^2 \, dt$$

as the sub-criterion to be minimized, where

$$\tau(t) \equiv t_f - t = \text{time-to-go at } t \qquad 2:3$$

Then:

if $r > 0$, resource expenditures early in the engagement are penalized more than expenditures late in the engagement

if $r = 0$, there is no time weighting of the guidance effort penalty

if $r < 0$, resource expenditures late in the engagement are penalized more than expenditures early in the engagement

Accordingly, we take

$$G_0 \equiv \tau^r \dot{N}^2 \qquad 2:4$$

Ultimately, means should be sought to select $r(t_i)$ as a function of the estimated uncertainty at t_i regarding the present states of the projectile and its target and, in particular, the predicted future states of the target. Further, $r(t_i)$ should be chosen, if possible, with rigorous consideration of the expected statistical moments of errors in these predictions.

Projectile and target state estimation and the identification of target maneuver strategies are not treated here, but these become very important topics as the work proceeds.

We do not now treat the case in which r changes with t after time t_i, because doing so significantly complicates the derivation.

2.2 Guidance Error Penalty

Consider a Cartesian inertial coordinate system in which the positions of objects are described by components x_1, x_2, and x_3. (See further discussion in the Appendix.) Then we may write a second-order Taylor series to express the predicted final position of the target for inertial axis k (k = 1, 2, 3):

$$\hat{x}_{T_k}(t_f) = x_{T_k}(t) + \tau \dot{x}_{T_k}(t) + \frac{\tau^2}{2}\left[\ddot{x}_{T_k}(t) + \hat{\Delta}\ddot{x}_{T_k}(t)\right] \qquad 2:5$$

where ( ) denotes an estimated or predicted value of a variable and $\Delta\ddot{x}_T$ is the effective (but not a priori known) increment to target acceleration that, treated as shown in Eq. 2:5, will cause $\hat{x}_{Tk}(t_f)$ to be an accurate estimate over the interval of prediction. Estimation of $\Delta\ddot{x}_T$, an important topic, will not be dealt with in this report. For the present, we take

$$\ddot{x}_T' \equiv \ddot{x}_T + \hat{\Delta}\ddot{x}_T \qquad 2:6$$

whence

$$\hat{x}_{Tk}(t_f) = x_{Tk}(t) + \tau \dot{x}_{Tk}(t) + \frac{\tau^2}{2} \ddot{x}_{Tk}'(t) \qquad 2:7$$

For the projectile

$$x_k(t_f) = x_k(t) + \tau \dot{x}_k(t) + \frac{\tau^2}{2} \ddot{x}_k(t) \qquad 2:8$$

provided the neglected higher-order terms sum to zero as t approaches t_f. In general, for large τ, the proviso is violated. Nevertheless, it is admissible to consider a simplistic predictive error function

$$e_k(t) \equiv x_{Tk}(t) - x_k(t) + \tau\left[\dot{x}_{Tk}(t) - \dot{x}_k(t)\right] + \frac{\tau^2}{2}\left[\ddot{x}_{Tk}'(t) - \ddot{x}_k(t)\right] \qquad 2:9$$

which says, given τ and measures of the target and projectile states, x_T, $\dot{x}_T$, x, $\dot{x}$; given an estimate of the effective target acceleration over the prediction interval (t, t_f); and given a particular value of projectile acceleration over the same interval, a simplistic prediction of miss distance along axis k is as stated (Eq. 2:9). We note that the simplistic predictive error function, e(t), is not the same as the expected final miss distance, which we intend to make essentially zero via a two-point boundary-value (TPBV) solution. Rather, e(t) is used to communicate to the optimization a penalty for not nulling the simplistic prediction at every time t ($t_i \le t \le t_f$).

There is an important difference between (a) use of the simplistic predictive error function as a sub-criterion in a variational TPBV solution and (b) conventional guidance methods that act to null some type of simplistic prediction throughout the engagement. The proposed optimum TPBV solution achieves specified end conditions and produces a trade-off between resource consumption and nulling of the simplistic prediction so as to achieve zero terminal error and minimize the expenditure of resources in the presence of uncertain information. In particular, this may be done for given levels of uncertainty about the target maneuver strategy. As limiting cases, for very large uncertainty, $\Delta\ddot{x}_T$ may be taken as zero, and for very little uncertainty, $\Delta\ddot{x}_T$ may be assumed to be known perfectly.

The predictive error penalty is introduced with the function

$$G_5 \equiv \tau^{-s} \cdot \sum_{k=1}^{3} e_k^2 \qquad 2:10$$

in which $s(t_i)$ has a role analogous to that of r, except that $s > 0$ corresponds to emphasis on the error penalty late in the engagement, and $s < 0$ pertains to emphasis early in the engagement.

2.3 Penalty on Loss of Kinetic Energy

The kinetic energy of the projectile, per unit of mass, is

$$\frac{1}{2} \sum_{k=1}^{3} \dot{x}_k^2$$

The time rate of change of kinetic energy is, therefore

$$\sum_{k=1}^{3} \dot{x}_k \ddot{x}_k$$

Thus

$$-\int_{t_i}^{t_f} \sum_{k=1}^{3} \dot{x}_k \ddot{x}_k \, dt$$

represents the loss in kinetic energy over the (t_i, t_f) interval. The greater this loss, the greater the sacrifice in range and maneuverability of the projectile. To penalize kinetic energy loss, one may introduce integrals of the following sub-criteria

$$G_1 \equiv -\dot{x}_1 \ddot{x}_1, \qquad G_2 \equiv -\dot{x}_2 \ddot{x}_2, \qquad G_3 \equiv -\dot{x}_3 \ddot{x}_3, \qquad 2:11$$

2.4 Limit on Total Number of Squibs Available

An approach toward limiting the total number of squibs fired is to introduce the penalty

$$G_4 \equiv N^2 \qquad 2:12$$

2.5 Distance-Travelled Constraints

It may be useful to maximize the distance travelled by the projectile. The downrange distance traversed is the integral of

$$G_6 \equiv \dot{x}_1 \sin \psi_i + \dot{x}_2 \cos \psi_i \qquad 2:13$$

and the crossrange distance is the integral of

$$G_7 \equiv \dot{x}_1 \cos \psi_i - \dot{x}_2 \sin \psi_i \qquad 2:14$$

where ψ_i is the velocity heading at time t_i. The change of altitude (not necessarily to be maximized) is the integral of

$$G_8 \equiv \dot{x}_3 \qquad 2:15$$

The total distance (slant range) traversed is the integral of

$$G_9 \equiv \frac{(x_1 - x_{1_i})\dot{x}_1 + (x_2 - x_{2_i})\dot{x}_2 + (x_3 - x_{3_i})\dot{x}_3}{\left[(x_1 - x_{1_i})^2 + (x_2 - x_{2_i})^2 + (x_3 - x_{3_i})^2\right]^{1/2}} \qquad 2:16$$

2.6 Anholonomic Constraints

From Appendix A and first principles, one has the following anholonomic constraint conditions:

$$f_1 \equiv m\dot{V} + mg \sin\gamma + qS C_{D_0} + (T^*)^2 (qS C_3 + C_0 H_1)\dot{N} = 0 \qquad 2:17$$

$$f_2 \equiv mV\dot{\psi} \cos\gamma + T^* (qS C_0 H_3 - \sin\varphi)\dot{N} = 0 \qquad 2:18$$

$$f_3 \equiv mV\dot{\gamma} + mg \cos\gamma + T^* (qS C_0 H_5 - \cos\varphi)\dot{N} = 0 \qquad 2:19$$

$$f_4 \equiv \dot{V} - U_1 \cos\gamma \sin\psi - U_2 \cos\gamma \cos\psi - U_3 \sin\gamma = 0 \qquad 2:20$$

$$f_5 \equiv V\dot{\gamma} + U_1 \sin\gamma \sin\psi + U_2 \sin\gamma \cos\psi - U_3 \cos\gamma = 0 \qquad 2:21$$

$$f_6 \equiv V\dot{\psi} \cos\gamma - U_1 \cos\psi + U_2 \sin\psi = 0 \qquad 2:22$$

$$f_7 \equiv V_1 - \dot{x}_1 = 0 \qquad 2:23$$

$$f_8 \equiv V_2 - \dot{x}_2 = 0 \qquad 2:24$$

$$f_9 \equiv V_3 - \dot{x}_3 = 0 \qquad 2:25$$

$$f_{10} \equiv U_1 - \dot{V}_1 = 0 \qquad 2:26$$

$$f_{11} \equiv U_2 - \dot{V}_2 = 0 \qquad 2:27$$

$$f_{12} \equiv U_3 - \dot{V}_3 = 0 \qquad 2:28$$

$$f_{13} \equiv V_1 - V \cos\gamma \sin\psi = 0 \qquad 2:29$$

$$f_{14} \equiv V_2 - V \cos\gamma \cos\psi = 0 \qquad 2:30$$

$$f_{15} \equiv V_3 - V \sin\gamma = 0 \qquad 2:31$$

where:

$$\varphi = Pt + \varphi_i \qquad 2:32$$

$$\dot{\varphi} = P \qquad 2:33$$

$$H_1 = 1 \qquad 2:34$$

$$H_3 \equiv - C_{N\alpha} \sin\varphi + C_2 \cos\varphi \qquad 2:35$$

$$\dot{H}_3 = - P\left(C_{N\alpha} \cos\varphi + C_2 \sin\varphi\right) \qquad 2:36$$

$$H_5 \equiv - C_{N\alpha} \cos\varphi - C_2 \sin\varphi \qquad 2:37$$

$$\dot{H}_5 = P\left(C_{N\alpha} \sin\varphi - C_2 \cos\varphi\right) \qquad 2:38$$

$$q \equiv \frac{1}{2}\rho V^2 \qquad 2:39$$

$$\dot{q} = \rho V\dot{V} \qquad 2:40$$

and the parameters assumed (for now) to be <u>constant</u> are:

m = mass
g = gravity acceleration
ρ = atmosphere density
S = reference area
C_{D0} = zero-normal-force value of drag coefficient
K = induced-drag factor (see Appendix B)
$C_{N\alpha}$ = normal force coefficient

$$C_0 \equiv \frac{1}{2}\left(\frac{1}{\omega_1} - \frac{1}{\omega_2}\right)\frac{l_T \exp\left(-\dfrac{\pi\zeta_1/2}{\sqrt{1-\zeta_1^2}}\right)}{I_{yy}\,\omega_1\sqrt{1-\zeta_1^2}} \qquad 2:41^*$$

*From Hutchings, Thomas D., Ref. 2, 1984, which defines the terminology in this factor.

$$C_2 \equiv C_{Y_{P\alpha}} d_{ref} \left(\frac{P}{2V}\right) = \text{normalized Magnus force coefficient} \qquad 2:42$$

$$C_3 \equiv K\left(C_{N_\alpha}{}^2 + C_2{}^2\right) C_0{}^2 \qquad 2:43$$

$$T^* = \text{scaled thrust} = T\left(\frac{\Delta t_{\text{squib burn}}}{\Delta t_{\text{decision interval}}}\right) = T\,\delta t/\Delta t << T \qquad 2:44$$

P = rolling rate

φ_i = roll attitude at t_i

3. DERIVATION OF GOVERNING EQUATIONS

3.1 Necessary and Sufficient Conditions of the Calculus of Variations

The calculus of variations necessary and sufficient conditions for existence of a minimizing function F(t) are as follows. For the zeroth-order freedoms of the solution, here represented by u:

$$F_u = 0 \qquad (t_i \le t \le t_f) \qquad \text{Euler-Lagrange} \qquad 3:1$$

$$F_{uu} \ge 0 \qquad (t_i \le t \le t_f) \qquad \text{Legendre} \qquad 3:2$$

and for the first-order freedoms, here denoted by x:

$$\dot{F}_{\dot{x}} = F_x \qquad (t_i \le t \le t_f) \qquad \text{Euler-Lagrange} \qquad 3:3$$

$$F_{\dot{x}}(t_f) = 0 \qquad 3:4$$

$$F_{\dot{x}\dot{x}} \ge 0 \qquad (t_i \le t \le t_f) \qquad \text{Legendre} \qquad 3:5$$

At least one partial derivative of the form of Ineqs. 3:2 and 3:5 must be positive, and all cross partials at the u and $\dot{x}$ levels (such as $F_{u_1u_2}$ and $F_{u_1\dot{x}_1}$) must be zero (Kirk, Ref. 4). Furthermore:

$$F(t_f) = 0 \qquad 3:6$$

3.2 Integrand of Performance Criterion

From Eqs. 2:2, 2:10, 2:11, and 2:12 - 2:15 (omitting 2:16 at this time), the integrand of the proposed performance criterion is

$$\begin{aligned} F \equiv\ & W_0 \tau^r \dot{N}^2 - \sum_{k=1}^{3} W_k \dot{x}_k \ddot{x}_k + W_4 N^2 + W_5 \tau^{-s} \sum_{k=1}^{3} e_k^2 \\ & + W_6 \left(\dot{x}_1 \sin \psi_i + \dot{x}_2 \cos \psi_i \right) + W_7 \left(\dot{x}_1 \cos \psi_i - \dot{x}_2 \sin \psi_i \right) \\ & + W_8 V \sin \gamma + \sum_{j=1}^{15} \lambda_j f_j \end{aligned} \qquad 3:7$$

Substituting Eqs. 2:9 and 2:23 - 2:28

$$F = W_0 \tau^r \dot{N}^2 - \sum_{k=1}^{3} W_k V_k U_k + W_4 N^2 + W_5 \tau^{-s} \sum_{k=1}^{3} \left[x_{Tk} - x_k + \tau\left(\dot{x}_{Tk} - V_k\right) + \frac{\tau^2}{2}\left(\ddot{x}_{Tk} - U_k\right)\right]^2 + W_6\left(V_1 \sin\psi_i + V_2 \cos\psi_i\right) + W_7\left(V_1 \cos\psi_i - V_2 \sin\psi_i\right) + W_8 V_3 + \sum_{j=1}^{15} \lambda_j f_j \qquad 3:8$$

The variational freedoms are:

Zeroth–order
U_1, U_2, U_3

First–order
$V_1, V_2, V_3, x_1, x_2, x_3, V, \psi, \gamma, N$

3.3 Derivation

Eq. 3.1 for the zeroth–order freedoms applied to the integrand 3:8 yields:

$$\frac{1}{2} W_5 \tau^{-s+4} U_1 = W_1 V_1 + W_5 \tau^{-s+2}\left[x_{T1} - x_1 + \tau\left(\dot{x}_{T1} - V_1\right) + \frac{\tau^2}{2}\ddot{x}_{T1}\right] + \left(\lambda_4 \cos\gamma - \lambda_5 \sin\gamma\right)\sin\psi + \lambda_6 \cos\psi - \lambda_{10} \qquad 3:9$$

$$\frac{1}{2} W_5 \tau^{-s+4} U_2 = W_2 V_2 + W_5 \tau^{-s+2}\left[x_{T2} - x_2 + \tau\left(\dot{x}_{T2} - V_2\right) + \frac{\tau^2}{2}\ddot{x}_{T2}\right] + \left(\lambda_4 \cos\gamma - \lambda_5 \sin\gamma\right)\cos\psi - \lambda_6 \sin\psi - \lambda_{11} \qquad 3:10$$

$$\frac{1}{2} W_5 \tau^{-s+4} U_3 = W_3 V_3 + W_5 \tau^{-s+2} \left[x_{T3} - x_3 + \tau \left(\dot{x}_{T3} - V_3 \right) + \frac{\tau^2}{2} \ddot{x}_{T3} \right] + \lambda_4 \sin\gamma + \lambda_5 \cos\gamma - \lambda_{12} \qquad 3:11$$

Ineq. 3:2 for the zeroth–order freedoms provides:

$$\frac{1}{2} W_5 \tau^{-s+4} \geq 0 \qquad 3:12$$

which is satisfied provided W_5 is non–negative.

Eq. 3:3 applied for the V_1, V_2, V_3 freedoms leads to:

$$\dot{\lambda}_{10} = W_1 U_1 + 2W_5 \tau^{-s+1} e_1 - W_6 \sin\psi_i - W_7 \cos\psi_i - \lambda_7 - \lambda_{13} \qquad 3:13$$

$$\dot{\lambda}_{11} = W_2 U_2 + 2W_5 \tau^{-s+1} e_2 - W_6 \cos\psi_i + W_7 \sin\psi_i - \lambda_8 - \lambda_{14} \qquad 3:14$$

$$\dot{\lambda}_{12} = W_3 U_3 + 2W_5 \tau^{-s+1} e_3 + W_8 - \lambda_9 - \lambda_{15} \qquad 3:15$$

where e_k ($k = 1, 2, 3$) is given by Eq. 2:9.

From Eq. 3:4 applied for V_1, V_2, V_3:

$$\lambda_{10f} = \lambda_{11f} = \lambda_{12f} = 0 \qquad 3:16$$

Ineq. 3:5 gives:

$$F \dot{V}_1 \dot{V}_1 = F \dot{V}_2 \dot{V}_2 = F \dot{V}_3 \dot{V}_3 = 0 \qquad 3:17$$

and therefore these conditions are always satisfied.

Next, Eq. 3:3 applied for x_1, x_2, x_3 yields:

$$\dot{\lambda}_7 = 2W_5 \tau^{-s} e_1 \qquad 3:18$$

$$\dot{\lambda}_8 = 2W_5 \tau^{-s} e_2 \qquad 3:19$$

$$\dot{\lambda}_9 = 2W_5 \tau^{-s} e_3 \qquad 3:20$$

and from Eq. 3:4:

$$\lambda_{7f} = \lambda_{8f} = \lambda_{9f} = 0 \qquad 3:21$$

Ineq. 3:5 becomes:

$$F_{\dot{x}_1\dot{x}_1} = F_{\dot{x}_2\dot{x}_2} = F_{\dot{x}_3\dot{x}_3} = 0 \qquad 3:22$$

For the V, ψ, and γ freedoms, Eq. 3:3 produces:

$$\begin{aligned}\dot{\lambda}_1 m + \dot{\lambda}_4 = \rho VS \Big\{ & \lambda_1 \left[C_{D_0} + (T^*)^2 C_3 \dot{N} \right] + T^* \left(\lambda_2 H_3 + \lambda_3 H_5 \right) C_0 \dot{N} \Big\} \\ & + \left(\lambda_2 m + \lambda_6 \right) \dot{\psi} \cos\gamma + \left(\lambda_3 m + \lambda_5 \right) \dot{\gamma} \\ & - \left(\lambda_{13} \sin\psi + \lambda_{14} \cos\psi \right) \cos\gamma - \lambda_{15} \sin\gamma \end{aligned} \qquad 3:23$$

$$\begin{aligned}\dot{\lambda}_2 m + \dot{\lambda}_6 = \frac{1}{V \cos\gamma} \Big[& \left(\lambda_2 m + \lambda_6 \right) \left(V\dot{\gamma} \sin\gamma - \dot{V} \cos\gamma \right) \\ & - \left(\lambda_4 \cos\gamma - \lambda_5 \sin\gamma \right) \left(U_1 \cos\psi - U_2 \sin\psi \right) \\ & + \lambda_6 \left(U_1 \sin\psi + U_2 \cos\psi \right) \Big] - \lambda_{13} \cos\psi + \lambda_{14} \sin\psi \end{aligned} \qquad 3:24$$

$$\begin{aligned}\dot{\lambda}_3 m + \dot{\lambda}_5 = \frac{1}{V} \Big[& - \left(\lambda_3 m + \lambda_5 \right) \dot{V} + mg \left(\lambda_1 \cos\gamma - \lambda_3 \sin\gamma \right) \\ & - \left(\lambda_2 m + \lambda_6 \right) V \dot{\psi} \sin\gamma + \left(\lambda_4 \cos\gamma + \lambda_5 \sin\gamma \right) U_3 \\ & + \left(\lambda_4 \sin\gamma + \lambda_5 \cos\gamma \right) \left(U_1 \sin\psi + U_2 \cos\psi \right) \Big] \\ & - \left(\lambda_{13} \sin\psi + \lambda_{14} \cos\psi \right) \sin\gamma - \lambda_{15} \cos\gamma \end{aligned} \qquad 3:25$$

Typically, $\dot{\lambda}_4$, $\dot{\lambda}_5$, and $\dot{\lambda}_6$ are taken as zero from t_i to t_f. (See discussion in Section 4.)

Eqs. 3:23 - 3:25 involve $\dot{V}$, $\dot{\psi}$, and $\dot{\gamma}$, which are functions of $\dot{N}$ (see Eqs. 2:17 - 2:19), and U_1, U_2, U_3, which, in this case, are the "command" values of projectile accelerations governed by the variational relationships in Eqs. 3:9 - 3:11. Thus:

$$\dot{\lambda}_1 = A_{10} + A_{11}\dot{N} \qquad 3:26$$

$$\dot{\lambda}_2 = A_{20} + A_{21}\dot{N} \qquad 3:27$$

$$\dot{\lambda}_3 = A_{30} + A_{31}\dot{N} \qquad 3:28$$

in which:

$$A_{10} \equiv \frac{1}{m}\left[\rho V S \lambda_1 C_{D_0} - \left(\lambda_3 m + \lambda_5\right)\frac{g\cos\gamma}{V} - \left(\lambda_{13}\sin\psi + \lambda_{14}\cos\psi\right)\cos\gamma - \lambda_{15}\sin\gamma\right] \qquad 3:29$$

$$A_{11} \equiv \frac{T^*}{m^2 V}\left\{2qSm\left[\lambda_1 T^* C_3 + C_0\left(\lambda_2 H_3 + \lambda_3 H_5\right)\right] - \left(\lambda_2 m + \lambda_6\right)\left(qS C_0 H_3 - \sin\varphi\right) - \left(\lambda_3 m + \lambda_5\right)\left(qS C_0 H_5 - \cos\varphi\right)\right\} \qquad 3:30$$

$$A_{20} \equiv \frac{1}{m}\left[\left(\lambda_2 m + \lambda_6\right)\frac{qS C_{D_0}}{m V} + \left(\lambda_5\sin\gamma - \lambda_4\cos\gamma\right)\left(U_1\cos\psi - U_2\sin\psi\right) + \lambda_6\left(U_1\sin\psi + U_2\cos\psi\right) - \lambda_{13}\cos\psi + \lambda_{14}\sin\psi\right] \qquad 3:31$$

$$A_{21} \equiv \frac{T^*}{m^2 V\cos\gamma}\left(\lambda_2 m + \lambda_6\right)\left[T^*\left(qS C_3 + C_0 H_1\right)\cos\gamma - \left(qS C_0 H_5 - \cos\varphi\right)\sin\gamma\right] \qquad 3:32$$

$$A_{30} \equiv \frac{1}{m}\Big[\big(\lambda_3 m + \lambda_5\big)\big(g\sin\gamma + qS\,C_{D_0}/m\big)/V + mg\,\big(\lambda_1\cos\gamma - \lambda_3\sin\gamma\big)/V + \big(\lambda_4\sin\gamma + \lambda_5\cos\gamma\big)\big(U_1\sin\psi + U_2\cos\psi\big)/V + \big(\lambda_5\sin\gamma - \lambda_4\cos\gamma\big)U_3/V + \big(\lambda_{13}\sin\psi + \lambda_{14}\cos\psi\big)\sin\gamma - \lambda_{15}\cos\gamma\Big] \qquad 3:33$$

$$A_{31} \equiv \frac{T^*}{m^2 V}\Big[\big(\lambda_3 m + \lambda_5\big)T^*\,(qS\,C_3 + C_0\,H_1) + \big(\lambda_2 m + \lambda_6\big)\big(qS\,C_0\,H_3 - \sin\varphi\big)\tan\gamma\Big] \qquad 3:34$$

where:

$$U_1 = 2\Big[W_1 V_1 + \big(\lambda_4\cos\gamma - \lambda_5\sin\gamma\big)\sin\psi + \lambda_6\cos\psi - \lambda_{10}\Big]\Big/\big(W_5\,\tau^{-S} + 4\big) + 2\Big[x_{T_1} - x_1 + \tau\,(\dot{x}_{T_1} - V_1) + \frac{\tau^2}{2}\ddot{x}'_{T_1}\Big]/\tau^2 \qquad 3:35$$

$$U_2 = 2\Big[W_2 V_2 + \big(\lambda_4\cos\gamma - \lambda_5\sin\gamma\big)\cos\psi - \lambda_6\sin\psi - \lambda_{11}\Big]\Big/\big(W_5\,\tau^{-S} + 4\big) + 2\Big[x_{T_2} - x_2 + \tau\,(\dot{x}_{T_2} - V_2) + \frac{\tau^2}{2}\ddot{x}'_{T_2}\Big]/\tau^2 \qquad 3:36$$

$$U_3 = 2\big(W_3 V_3 + \lambda_4\sin\gamma + \lambda_5\cos\gamma - \lambda_{12}\big)\big/\big(W_5\,\tau^{-S} + 4\big) + 2\Big[x_{T_3} - x_3 + \tau\,(\dot{x}_{T_3} - V_3) + \frac{\tau^2}{2}\ddot{x}'_{T_3}\Big]/\tau^2 \qquad 3:37$$

and where (see Eqs. 3:13 - 3:15 and 3:18 - 3:20):

$$\dot{\lambda}_{10} = W_1 U_1 + \tau\dot{\lambda}_7 - \Lambda_7 \qquad 3:38$$

$$\dot{\lambda}_{11} = W_2 U_2 + \tau\dot{\lambda}_8 - \Lambda_8 \qquad 3:39$$

$$\dot{\lambda}_{12} = W_3 U_3 + \tau\dot{\lambda}_9 - \Lambda_9 \qquad 3:40$$

which uses the definitions:

$$\Lambda_7 \equiv \lambda_7 + \lambda_{13} + W_6 \sin\psi_i + W_7 \cos\psi_i \qquad 3:41$$

$$\Lambda_8 \equiv \lambda_8 + \lambda_{14} + W_6 \cos\psi_i + W_7 \sin\psi_i \qquad 3:42$$

$$\Lambda_9 \equiv \lambda_9 + \lambda_{15} - W_8 \qquad 3:43$$

From Eq. 3:4:

$$\lambda_{1f}\, m + \lambda_{4f} = 0 \qquad 3:44$$

$$\left(\lambda_{2f}\, m + \lambda_{6f}\right) V_f \cos\gamma_f = 0 \qquad 3:45$$

$$\left(\lambda_{3f}\, m + \lambda_{5f}\right) V_f = 0 \qquad 3:46$$

whence:

$$\lambda_{1f} = -\lambda_{4f}/m \qquad 3:47$$

$$\lambda_{2f} = -\lambda_{6f}/m \qquad 3:48$$

$$\lambda_{3f} = -\lambda_{5f}/m \qquad 3:49$$

Continuing, for the N freedom

$$F_N = 2W_4 N \qquad 3:50$$

and

$$F_{\dot{N}} = 2\, W_0\, \tau^r\, \dot{N} + \lambda_1 (T^*)^2 (qS\, C_3 + C_0 H_1) + \lambda_2 T^* \left(qS\, C_0 H_3 - \sin\varphi\right) + \lambda_3 T^* \left(qS\, C_0 H_5 - \cos\varphi\right) \qquad 3:51$$

$$\dot{F}_{\dot{N}} = 2W_0\left(\tau^r \dddot{N} - r\tau^{r-1}\ddot{N}\right) + \dot{\lambda}_1 (T^*)^2 (qSC_3 + C_0 H_1) + \lambda_1 (T^*)^2 (\dot{q}SC_3 + C_0 \dot{H}_1) + \dot{\lambda}_2 T^*\left(qSC_0 H_3 - \sin\varphi\right) + \lambda_2 T^*\left(\dot{q}SC_0 H_3 + qSC_0 \dot{H}_3 - P\cos\varphi\right) + \dot{\lambda}_3 T^*\left(qSC_0 H_5 + \cos\varphi\right) + \lambda_3 T^*\left(\dot{q}SC_0 H_5 + qSC_0 \dot{H}_5 + P\sin\varphi\right) \quad 3:52$$

and Eq. 3:3 yields, upon substitution of Eqs. 2:17, 2:40, and 3:26 - 3:28:

$$A_2\ddot{N} + A_1\dot{N} + A_0 N = B \quad 3:53$$

where:

$$A_2 \equiv 2W_0\tau^r \quad 3:54$$

$$A_1 \equiv -2W_0 r\tau^{r-1} \quad 3:55^*$$

$$A_0 \equiv -2W_4 \quad 3:56$$

$$B \equiv -T^*\Big[T^*(qSC_3 + C_0 H_1)A_{10} + \left(qSC_0 H_3 - \sin\varphi\right)A_{20} + \left(qSC_0 H_5 - \cos\varphi\right)A_{30} - \rho VS\left(\lambda_1 T^* C_3 + \lambda_2 C_0 H_3 + \lambda_3 C_0 H_5\right)\left(g\sin\gamma + qSC_{D_0}/m\right) + \lambda_2 P\left(qSC_0 H_5 - \cos\varphi\right) - \lambda_3 P\left(qSC_0 H_3 - \sin\varphi\right)\Big] \quad 3:57$$

The derivative $\dddot{N}$ is zero except at a finite number of points of singularity. Eq. 3:53 must be satisfied between these singular points, i.e., for the open interval $(t_i, t_i + \Delta t)$ and succeeding decision intervals. Thus

$$\dot{N} = (B - A_0 N)/A_1 \quad 3:58$$

From Eqs. 3:4, 3:47 - 3:49, and 3:51:

---

* Note that all other terms in $\dot{N}$ that appear in Eq. 3:52 cancel one another.

$$\lambda_{4f} = \frac{-\lambda_{6f}\left(q_f S C_0 H_{3f} - \sin\varphi_f\right) - \lambda_{5f}\left(q_f S C_0 H_{5f} - \cos\varphi_f\right) - 2W_0\tau_f^r \dot{N}_f\, m/T^*}{T^*\left(q_f S C_3 + C_0 H_{1f}\right)} \qquad 3:59$$

From Ineq. 3:5:

$$A_2 = 2W_0\tau^r \geq 0 \qquad 3:60$$

which is always satisfied if $W_0 > 0$.

It can readily be shown that all cross partials at the "u" and "x" levels are zero for the subject performance criterion.

Lastly, the necessary condition of Eq. 3:6 provides (note that the f_j's are zero in Eq. 3:8):

$$W_0\tau_f^r \dot{N}_f^2 - \sum_{k=1}^{3} W_k V_{kf} U_{kf} + W_4 N_f{}^2 + W_5 t_f^{-s} \sum_{k=1}^{3} e_{kf}^2 + W_6\left(V_{1f}\sin\psi_i + V_{2f}\cos\psi_i\right) + W_7\left(V_{1f}\cos\psi_i - V_{2f}\sin\psi_i\right) + W_8 V_{3f} = 0 \qquad 3:61$$

Thus, taking the desired range of s, $s > 0$, requires that $\sum_{k=1}^{3} e_{kf}^2$ approach zero more quickly than τ_f^{-s}, namely

$$\text{(i)} \quad \lim_{t \to t_f} \frac{\sum_{k=1}^{3} e_k{}^2}{\tau^s} = 0 \qquad 3:62$$

In addition, if $r > 0$

$$\text{(ii)} \quad -\sum_{k=1}^{3} W_k V_{kf} U_{kf} + W_4 N_f{}^2 + W_6\left(V_{1f}\sin\psi_i + V_{2f}\cos\psi_i\right) + W_7\left(V_{1f}\cos\psi_i - V_{2f}\sin\psi_i\right) + W_8 V_{3f} = 0 \qquad 3:63$$

If $r < 0$, it is necessary to satisfy 3:62 and 3:63 as well as

(iii) $\dot{N}_f = 0$ 3:64

In the case where $r = 0$, it is necessary to satisfy 3:62 as well as

$$\text{(iv)}\ W_0 \dot{N}_f^2 - \sum_{k=1}^{3} W_k V_{kf} U_{kf} + W_4 N_f^2 + W_6 \left(V_{1f} \sin \psi_i + V_{2f} \cos \psi_i \right) + W_7 \left(V_{1f} \cos \psi_i - V_{2f} \sin \psi_i \right) + W_8 V_{3f} = 0 \quad 3:65$$

Solution of the governing equations is discussed in the next section.

4. SOLVING THE GOVERNING EQUATIONS

4.1 Establishing the Final Velocity

To establish the final velocity for off-line reverse integration (data base generation), if $r > 0$, Eqs. 3:63 and 3:65 require that

$$V_{1f}\left(W_1 \dot{V}_{1f} - W_6 \sin\psi_i - W_7 \cos\psi_i\right) + V_{2f}\left(W_2 \dot{V}_{2f} - W_6 \cos\psi_i + W_7 \sin\psi_i\right) + V_{3f}\left(W_3 \dot{V}_{3f} - W_8\right) - W_4 N_f^2 = 0 \qquad 4:1$$

Eqs. 3:1 and 3:9 - 3:11 become , at t_f:

$$W_1 V_{1f} = -\left(\lambda_{4f} \cos\gamma_f - \lambda_{5f} \sin\gamma_f\right) \sin\psi_f - \lambda_{6f} \cos\psi_f + \lambda_{10f} \qquad 4:2$$

$$W_2 V_{2f} = -\left(\lambda_{4f} \cos\gamma_f - \lambda_{5f} \sin\gamma_f\right) \cos\psi_f + \lambda_{6f} \sin\psi_f + \lambda_{11f} \qquad 4:3$$

$$W_3 V_{3f} = -\lambda_{4f} \sin\gamma_f - \lambda_{5f} \cos\gamma_f + \lambda_{12f} \qquad 4:4$$

However, $\lambda_{10f} = \lambda_{11f} = \lambda_{12f} = 0$ (Eq. 3:16), and if W_1, W_2, W_3 are not zero, Eqs. 4:1 - 4:4 provide

$$\left[\left(\lambda_{4f} \cos\gamma_f - \lambda_{5f} \sin\gamma_f\right) \sin\psi_f + \lambda_{6f} \cos\psi_f\right] \left(\dot{V}_{1f} - W_{61} \sin\psi_i - W_{71} \cos\psi_i\right) + \left[\left(\lambda_{4f} \cos\gamma_f - \lambda_{5f} \sin\gamma_f\right) \cos\psi_f - \lambda_{6f} \sin\psi_f\right] \left(\dot{V}_{2f} - W_{62} \cos\psi_i + W_{72} \sin\psi_i\right) + \left(\lambda_{4f} \sin\gamma_f + \lambda_{5f} \cos\gamma_f\right) \left(\dot{V}_{3f} - W_{83}\right) + W_4 N_f^2 = 0 \qquad 4:5$$

where:

$$W_{61} = W_6/W_1,\ W_{71} = W_7/W_1,\ W_{62} = W_6/W_2,\ W_{72} = W_7/W_2,\ W_{83} = W_8/W_3 \qquad 4:6$$

Moreover, the inverse forms of Eqs. 2:20 - 2:22 are:

$$\dot{V}_1 = U_1 = (\dot{V}\cos\gamma - V\dot{\gamma}\sin\gamma)\sin\psi + V\dot{\psi}\cos\gamma\cos\psi \qquad 4:7$$

$$\dot{V}_2 = U_2 = (\dot{V}\cos\gamma - V\dot{\gamma}\sin\gamma)\cos\psi - V\dot{\psi}\cos\gamma\sin\psi \qquad 4:8$$

$$\dot{V}_3 = U_3 = \dot{V}\sin\gamma + V\dot{\gamma}\cos\gamma \qquad 4:9$$

If $N = 0$, Eqs. A:22 - A:24 provide:

$$\dot{V} = -g\sin\gamma - qS C_{D_0}/m \qquad 4:10$$

$$\dot{\psi} = 0 \qquad 4:11$$

$$\dot{\gamma} = -\frac{g\cos\gamma}{V} \qquad 4:12$$

and Eqs. 4:7 - 4:9 become:

$$\left.\begin{aligned} \dot{V}_1 &= -q\,(S C_{D_0}/m)\cos\gamma\sin\psi \\ \dot{V}_2 &= -q\,(S C_{D_0}/m)\cos\gamma\cos\psi \\ \dot{V}_3 &= -g - q\,(S C_{D_0}/m)\sin\gamma \end{aligned}\right\} \quad (\text{if } \dot{N} = 0) \qquad 4:13$$

whence Eq. 4:5 yields

$$\begin{aligned} & q_f\,(S C_{D_0}/m)\left(J_{1f}\cos\gamma_f\sin\psi_f + J_{2f}\cos\gamma_f\cos\psi_f + J_{3f}\sin\gamma_f\right) \\ & \quad + J_{1f}\left(W_{61}\sin\psi_i + W_{71}\cos\psi_i\right) + J_{2f}\left(W_{62}\cos\psi_i - W_{72}\sin\psi_i\right) \\ & \quad + J_{3f}\left(W_{83} + g\right) - W_4 N_f^2 = 0 \end{aligned} \qquad 4:14$$

where:

$$J_{1f} \equiv \left(\lambda_{4f}\cos\gamma_f - \lambda_{5f}\sin\gamma_f\right)\sin\psi_f + \lambda_{6f}\cos\psi_f \qquad 4:15$$

$$J_{2f} \equiv \left(\lambda_{4f} \cos \gamma_f - \lambda_{5f} \sin \gamma_f\right) \cos \psi_f - \lambda_{6f} \sin \psi_f \qquad 4:16$$

$$J_{3f} \equiv \lambda_{4f} \sin \gamma_f + \lambda_{5f} \cos \gamma_f \qquad 4:17$$

Now, substituting Eq. 2:34, 2:35, and 2:37 into Eq. 3:59, one obtains, if $\dot{N}_f = 0$ and/or if $r > 0$,

$$\lambda_{4f} = \frac{\left[\lambda_{5f} q_f S C_0 C_2 + \lambda_{6f}\left(q_f S C_0 C_{N\alpha} + 1\right)\right] \sin \varphi_f}{T^*\left(q_f S C_3 + C_0\right)} + \frac{\left[\lambda_{5f}\left(q_f S C_0 C_{N\alpha} + 1\right) - \lambda_{6f} q_f S C_0 C_2\right] \cos \varphi_f}{T^*\left(q_f S C_3 + C_0\right)} \qquad 4:18$$

For the case $\varphi_f = 0$, Eq. 4:18 becomes:

$$\lambda_{4f} = \frac{\lambda_{5f} K_{5f} - \lambda_{6f} K_{6f}}{K_{0f}} \qquad 4:19$$

in which:

$$K_{5f} \equiv q_f S C_0 C_{N\alpha} + 1 \qquad 4:20$$

$$K_{6f} \equiv q_f S C_0 C_2 \qquad 4:21$$

$$K_{7f} \equiv T^*\left(q_f S C_3 + C_0\right) \qquad 4:22$$

whence:

$$J_{1f} = \left(\lambda_{5f} \sin \psi_f\right)\left(\frac{K_{5f}}{K_{0f}} \cos \gamma_f - \sin \gamma_f\right) + \lambda_{6f}\left(\cos \gamma_f - \frac{K_{6f}}{K_{0f}} \cos \gamma_f \sin \psi_f\right) \qquad 4:23$$

$$J_{2f} = \left(\lambda_{5f} \cos \psi_f\right)\left(\frac{K_{5f}}{K_{0f}} \cos \gamma_f - \sin \gamma_f\right) - \lambda_{6f}\left(\sin \psi_f + \frac{K_{6f}}{K_{0f}} \cos \gamma_f \cos \psi_f\right) \quad 4:24$$

$$J_{3f} = \lambda_{5f}\left(\frac{K_{5f}}{K_{0f}} \sin \gamma_f + \cos \gamma_f\right) - \lambda_{6f} \frac{K_{6f}}{K_{0f}} \sin \gamma_f \quad 4:25$$

and Eq. 4:14 takes the form

$$\begin{aligned}
&\left[\left(\lambda_{5f} \sin \psi_f\right)\left(K_{5f} \cos \gamma_f - K_{0f} \sin \gamma_f\right)\right. \\
&\left. + \lambda_{6f}\left(K_{0f} \cos \psi_f - K_{6f} \cos \gamma_f \sin \psi_f\right)\right]\left[q_f (S C_{D_0}/m) \cos \gamma_f \sin \psi_f\right. \\
&\qquad \left. + W_{61} \sin \psi_i + W_{71} \cos \psi_i\right] \\
&+ \left[\left(\lambda_{5f} \cos \psi_f\right)\left(K_{5f} \cos \gamma_f - K_{0f} \sin \gamma_f\right)\right. \\
&\left. - \lambda_{6f}\left(K_{0f} \sin \psi_f + K_{6f} \cos \gamma_f \cos \psi_f\right)\right]\left[q_f (S C_{D_0}/m) \cos \gamma_f \cos \psi_f\right. \\
&\qquad \left. + W_{62} \cos \psi_i - W_{72} \sin \psi_i\right] \\
&+ \left[\lambda_{5f}\left(K_{5f} \sin \gamma_f + K_{0f} \cos \gamma_f\right)\right. \\
&\qquad \left. - \lambda_{6f} K_{6f} \sin \gamma_f\right]\left[q_f (S C_{D_0}/m) \sin \gamma_f + W_{83} + g\right] \\
&- K_{0f} W_4 N_f^2 = 0
\end{aligned} \quad 4:26$$

Thus, from Eq. 4:26, defining

$$Q_f \equiv q_f S \quad 4:27$$

one has (if $\dot{N}_f = 0$, if $r > 0$, and if $\psi_f = \varphi_f = 0$)

$$a_f Q_f^2 + b_f Q_f + c_f = 0 \quad 4:28$$

where:

$$a_f \equiv C_0 (C_{D0}/m) \left(\lambda_{5f} C_{N\alpha} - \lambda_{6f} C_2\right) \qquad 4:29$$

$$\begin{aligned} b_f \equiv\ & \left(\lambda_{5f} C_{N\alpha} - \lambda_{6f} C_2\right) C_0 (W_{83} + g) \sin\gamma_f + \lambda_{5f} C_{D0}/m \\ & + T^* C_3 \left[\lambda_{5f}(W_{83} + g)\cos\gamma_f - W_4 N_f^2\right] \\ & + \left(W_{61}\sin\psi_i + W_{71}\cos\psi_i\right)\left[\lambda_{5f}\left(C_0 C_{N\alpha}\cos\gamma_f - T^* C_3 \sin\gamma_f\right)\sin\psi_f\right. \\ & \qquad \left. - \lambda_{6f}\left(C_0 C_2 \cos\gamma_f \sin\psi_f - T^* C_3 \cos\psi_f\right)\right] \\ & + \left(W_{62}\cos\psi_i - W_{72}\sin\psi_i\right)\left[\lambda_{5f}\left(C_0 C_{N\alpha}\cos\gamma_f - T^* C_3 \sin\gamma_f\right)\cos\psi_f\right. \\ & \qquad \left. - \lambda_{6f}\left(C_0 C_2 \cos\gamma_f \cos\psi_f + T^* C_3 \sin\psi_f\right)\right] \end{aligned} \qquad 4:30$$

$$\begin{aligned} c_f \equiv\ & \lambda_{5f}(W_{83} + g)\sin\gamma_f + T^* C_0 \left[\lambda_{5f}(W_{83} + g)\cos\gamma_f - W_4 N_f^2\right] \\ & + \left(W_{61}\sin\psi_i + W_{71}\cos\psi_i\right)\left[\lambda_{5f}\left(\cos\gamma_f - T^* C_0 \sin\gamma_f\right)\sin\psi_f\right. \\ & \qquad \left. + \lambda_{6f} T^* C_0 \cos\psi_f\right] \\ & + \left(W_{62}\cos\psi_i - W_{72}\sin\psi_i\right)\left[\lambda_{5f}\left(\cos\gamma_f - T^* C_0 \sin\gamma_f\right)\cos\psi_f\right. \\ & \qquad \left. - \lambda_{6f} T^* C_0 \sin\psi_f\right] \end{aligned} \qquad 4:31$$

Thus

$$Q_f = -\frac{b_f}{2a_f} - \left[\left(\frac{b_f}{2a_f}\right)^2 - \frac{c_f}{a_f}\right]^{1/2} \qquad (\text{if } \dot{N}_f = \dot{\psi}_f = \dot{\varphi}_f = 0) \qquad 4:32$$

If the minus sign is used ahead of the radical when λ_{5f}, W_6, W_7, W_8, and T^* are zero, one obtains $Q_f = -\left(mg \sin\gamma_f\right)/C_{D0}$, which requires that $\dot{V}_f = 0$ (see

Eq. 4:10), and this is not generally possible. Therefore, we use the plus sign. From Eqs. 4:27 and 4:32 it follows immediately that

$$V_f = \left(\frac{2Q_f}{\rho S}\right)^{1/2} \qquad 4:33$$

4.2 Velocity Transformations

The kinematic relationships between velocities expressed in the inertial (x_1, x_2, x_3) axes and wind (stability) axes are:

$$V = \left|(V_1^2 + V_2^2 + V_3^2)^{1/2}\right| \qquad 4:34$$

$$\gamma = \arcsin\left(V_3/V\right) \qquad 4:35$$

and if $V_2 \geq 0$

$$\psi = \arcsin\left(\frac{V_1}{V\cos\gamma}\right) \qquad 4:36$$

whereas if $V_2 < 0$

$$\psi = \pi - \arcsin\left(\frac{V_1}{V\cos\gamma}\right) \qquad 4:36$$

If $\gamma = \pm \pi/2$ (i.e., if $\cos\gamma = 0$), ψ is undefined. In Eq. 4:35, γ varies between $\pm \pi/2$, and in Eqs. 4:36, ψ ranges between $-\pi/2$ and $+3\pi/2$.

4.3 Estimation of Time-to-Go (τ)

A simplistic estimate of time-to-go is

$$\hat{\tau} = \left|\frac{\text{Range}}{\text{Closing Velocity}}\right| = \frac{\left|\Delta\underline{x}\right|}{-\left|\Delta\underline{V}\right|\cos\theta} = \frac{-\left|\sum_{k=1}^{3}\Delta x_k j_k\right|}{\left|\sum_{k=1}^{3}\Delta V_k j_k\right|\cos\theta} \qquad 4:37$$

where the j's are unitless direction vectors, θ is the angle between the vectors $\Delta\underline{x}$ and $\Delta\underline{V}$, and:

$$\Delta x_k = x_{Tk} - x_k \qquad 4:38$$

$$\Delta V_k = \dot{x}_{Tk} - V_k \qquad 4:39$$

As

$$\cos\theta = \frac{\Delta\underline{x}\ \Delta\underline{V}}{|\Delta\underline{x}||\Delta\underline{V}|} \qquad 4:40$$

$$\hat{\tau} = \frac{\sum_{k=1}^{3} \Delta x_k^2}{\sum_{k=1}^{3} \Delta x_k\ \Delta V_k} \qquad 4:41$$

Further discussion of to time-to-go estimation is presented in Appendix D.

4.4 Firing Commands for Multiple-Sector Thrusters

The CAT projectile may be designed with multiple (perhaps four) sectors of impulsive thrusters, each sector containing a number of squibs that fire through a common sector orifice. For any given sector there is an initial roll attitude, φ_{ij}, the index j denoting the sector. Correspondingly, Eqs. A:29 - A:30 and 2:35 - 2:38 provide multiple values of H_{3j}, $\dot{H}_{3j}$, H_{5j}, $\dot{H}_{5j}$ as functions of time. As a result, Eq. 3:30 provides a different $\dot{\lambda}_{1j}(t)$ value for each sector, and Eq. 3:57 and 3:58 give a different $\dot{N}_j(t)$ value for each sector.

The other equations of the OPTG solution are the same for all sectors.

4.5 Numerical Solution Procedure

To start numerical integration of the differential equations, the procedure is as follows:

(1) Describe projectile (m, S, C_{D0}, K, $C_{N\alpha}$, and lesser constants I_s, l_T, ζ_1, ω_1, ω_2, I_{yy}, d_{ref}, V_{av}, V_{muzzle}, N_{max}), environment (g, ρ), and target ($x_{T_1}(t)$, $x_{T_2}(t)$, $x_{T_3}(t)$).

(2) Specify guidance parameters (W_0,..., W_8, $r > 0$, $s > 0$).

(3) Specify P and Δt ($\Delta t \approx 0.05$ sec.). Compute firing constants T^*, C_0, C_2, and C_3 from Eqs. A:17, A:19, A:31 and A:32, respectively.

For Forward Integration

(4.F) Set $x_1 = x_2 = x_3 = 0$ and $N = 0$. Specify initial ψ, γ, and φ. Set $V = V_{muzzle}$. Specify λ_1,..., λ_{15} at t_0.

For Reverse Integration (Negative Δt)

(4.R.1) Set $x_1 = x_{T1f}$, $x_2 = x_{T2f}$, $x_3 = x_{T3f}$. Specify N_f, γ_f, λ_{5f}, λ_{6f}. Use ψ_f and φ_f of zero.

(4.R.2) Compute V_f from Eqs. 4:29 - 4:33, adjusting W_4, N_f, λ_5, and λ_6 if necessary to obtain acceptable V_f.

(4.R.3) Compute λ_{4f} from Eq. 4:18 and λ_{1f}, λ_{2f}, λ_{3f} from Eqs. 3:47 - 3:49. Set $\lambda_{7f} = \lambda_{8f} = \lambda_{9f} = \lambda_{10f} = \lambda_{11f} = \lambda_{12f} = 0$, per Eqs. 3:21 and 3:16.

Now the system states are initialized (or "finalized"), and the state derivatives may be calculated as follows:

(5) Compute V_1, V_2, V_3 from Eqs. 2:29 - 2:31. Compute e_1, e_2, e_3 from Eq. 2:9. Compute range from $((x_{T1} - x_1)^2 + (x_{T2} - x_2)^2 + (x_{T3} - x_3)^2)^{1/2}$.

(6) If range = 0, set $\hat{\tau} = 0$; otherwise compute $\hat{\tau}$ from Eq. 4:41.

(7) If $\hat{\tau} = 0$, compute U_1, U_2, U_3 (projectile accelerations) from the kinematic relationships, Eqs. 4:7 - 4:9; otherwise compute U_1, U_2, U_3 from the variational solution, Eqs. 3:35 - 3:37.

(8) If $\hat{\tau} = 0$, set $\dot{N}_f = 0$; otherwise, if unused squibs remain, compute $\dot{N}$ from Eqs. 3:55 - 3:58.

(9) If $\dot{N} >$ Threshold, set $\dot{N} = 1$. (Fire squib.)
If $\dot{N} \leq$ Threshold, set $\dot{N} = 0$. (Don't fire squib.)

(10) Compute wind (stability) axes rates $\dot{V}$, $\dot{\psi}$, $\dot{\gamma}$ from Eqs. A:33 - A:35.

(11) Compute $\dot{\lambda}_1$, $\dot{\lambda}_2$, $\dot{\lambda}_3$ from Eqs. 3:26 - 3:34.

(12) If $\hat{\tau} = 0$, set $\dot{\lambda}_{7f} = \dot{\lambda}_{8f} = \dot{\lambda}_{9f} = 0$; otherwise compute $\dot{\lambda}_7$, $\dot{\lambda}_8$, $\dot{\lambda}_9$ from Eqs. 3:18 - 3:20.

(13) Compute $\dot{\lambda}_{10}$, $\dot{\lambda}_{11}$, $\dot{\lambda}_{12}$, using Eqs. 3:38 - 3:40.

This completes the calculations of state derivatives from state values. Numerical integration may now be employed indefinitely to compute the states (φ, x_1, x_2, x_3, V, ψ, γ, and $\underline{\lambda}$) from their initial (or final) values and their respective derivatives (P, V_1, V_2, V_3, $\dot{V}$, $\dot{\psi}$, $\dot{\gamma}$, and $\dot{\underline{\lambda}}$).

4.6 Discussion

The best decision interval (Δt) is a function of projectile rolling rate (P). Preliminary evidence suggests that the difference $P - 1/\Delta t$ should be approximately + 1.0 Hz. This has the effect of providing roll attitude (φ)

segments of approximately 18 deg. between 20 discrete angles "available" for thruster firing, with each angle becoming available once per second.

The $\dot{N}$ command equation (Eq. 3:58) may be used in formal integration of the equations of motion and adjoint equations, preceeding from specified initial physical and adjoint (variable Lagrange multiplier) states, or it may be used in reverse integration from specified final (burst or impact) conditions. The use of reverse integration is helpful in mapping the behavior of the system as a function of the constant Lagrange multipliers (the W's), λ_5, and λ_6. When employing reverse integration, final values for the adjoint states are known from the variational conditions, except that λ_{5f} and λ_{6f} values must be stipulated from consideration of the required initial physical states. (Here we assume $\dot{\lambda}_4 = \dot{\lambda}_5 = \dot{\lambda}_6 = 0$.) We choose to refer to λ_5, λ_6 as the "steering" multipliers, because they are the principal mechanisms for satisfying the two-point boundary values imposed on guidance of the projectile. (The other λ's and the W's also influence the boundary values, but serve primarily to determine the optimal character of the solution. Furthermore, the final values of most of the other λ's and the $\dot{\lambda}$'s are known immediately once λ_{5f} and λ_{6f} have been provided.

In reverse integration, e_1, e_2, e_3 and $\dot{\lambda}_7$, $\dot{\lambda}_8$, $\dot{\lambda}_9$ begin at zero. In view of Eqs. 3:21, λ_7, λ_8, λ_9 also begin at zero. As long as e_1, e_2, e_3 remain zero, $\dot{\lambda}_{10}$, $\dot{\lambda}_{11}$, $\dot{\lambda}_{12}$ (ignoring W_1, W_2, W_3, W_6, W_7, and W_8) will be proportional to λ_{13}, λ_{14}, λ_{15}, respectively. Now, if $\dot{\lambda}_{10}$, $\dot{\lambda}_{11}$, $\dot{\lambda}_{12}$ are non-zero, λ_{10}, λ_{11}, λ_{12} will integrate away from their beginning zero values at t_f (Eqs. 3:16). As this occurs, U_1, U_2, U_3 will become biased, which is inappropriate to the purpose of maintaining e_1, e_2, e_3 at zero. We conclude, therefore, that it may be desirable that λ_{13}, λ_{14}, λ_{15} be kept equal to zero at all times.

The W_5 G_5 constraint, involving a simplistic prediction of final error, introduces the λ_7, λ_8, λ_9, λ_{10}, λ_{11}, λ_{12} adjoint differential equations, coupling them to the λ_1, λ_2, λ_3 adjoint differential equations via the variational (i.e., the "command" as contrasted with the kinematic) expressions for U_1, U_2, U_3. As $W_5 \rightarrow 0$, the coupling becomes weaker, disappearing altogther at $W_5 = 0$.

If $W_5 = 0$, the $\lambda_4, \lambda_5, \lambda_6, \lambda_{10}, \lambda_{11}$, and λ_{12} differential equations may, in principle, be eliminated from the solution, with the present roles of λ_4, λ_5, and λ_6 then assumed by $\lambda_{13}, \lambda_{15}$, and λ_{14}.

Two configurations of the operational OPTG guidance should be considered. In the first, polynomial networks would be used to estimate the initial values of all the λ's and to re-initialize the solution during the flight of the projectile. For example, before the gun is fired, the fire control system (FCS) could use polynomial networks to compute initial values of $\lambda_1, \lambda_2, \lambda_3$, with the initial values of $\lambda_4, ..., \lambda_{15}$ being the same for all trajectories and therefore not requiring pre-fire estimation. Then, with the projectile in flight, other polynomial networks in the FCS could be used to modify λ_5 and λ_6 to correct for target maneuvers and atmospheric disturbances of the projectile trajectory. In the second formulation, polynomial networks would provide the N decision (squib fire/no-fire decision). These configurations will be described further in future reports.

The OPTG formulation, implemented via either of the above configurations, will aid the system as it deals with information uncertainties and information denial and will provide efficient use of the projectile impulse thrusters. Regarding the problem of uncertainties, the OPTG guidance, implemented using APNs, will (a) respond optimally to predictable maneuvers of the target and (b) optimally desensitize the guidance to the adverse effects of unpredictable target motions and tracking errors. These benefits will accrue because, with careful simulation and data base generation for APN synthesis, the true non-parametric statistics of the problem will be mastered. (Incidentally, the payoff for good simulations will be measurably increased.) Regarding the efficient management of projectile thrust impulse, the calculus of variations will provide a data base of variational extremal trajectories that optimize the usage of projectile impulse, correcting by the most appropriate numbers of squib firings as the flights progress. Conventional guidance laws do not have the capability (the intelligence) to do this.

5. A BASELINE GUIDANCE LAW

If the simplistic predictive error function of Eq. 2:9 is set to zero, one obtains

$$U_{Ck} = \frac{2}{\tau^2}\left[x_{Tk} - x_k + \tau\left(x_{Tk} - \dot{x}_k\right) + \frac{\tau^2}{2}\ddot{x}_{Tk}\right] \qquad 5:1$$

in which the subscript "c" denotes a command value. Thus, from Eqs. 2:16 - 2:18:

$$\dot{V}_c = U_{c_1}\cos\gamma\sin\psi + U_{c_2}\cos\gamma\cos\psi + U_{c_3}\sin\gamma \qquad 5:2$$

$$(V\dot{\gamma})_c = -U_{c_1}\sin\gamma\sin\psi - U_{c_2}\sin\gamma\cos\psi + U_{c_3}\cos\gamma \qquad 5:3$$

$$(V\dot{\psi}\cos\gamma)_c = U_{c_1}\cos\psi - U_{c_2}\sin\psi \qquad 5:4$$

Eqs. A:33 - A:35 now become:

$$\dot{N}_c^{(1)} = \frac{K^*\left(\dot{V}_c + g\sin\gamma + qS\,C_{D_0}\right)}{(T^*)\left(qS\,C_3 + C_0\,H_1\right)} \qquad 5:5$$

$$\dot{N}_c^{(2)} = \frac{K^*\left(V\dot{\psi}\cos\gamma\right)_c}{qS\,C_0\,H_3 - \sin\varphi} \qquad 5:6$$

$$\dot{N}_c^{(3)} = \frac{K^*\left[(V\dot{\gamma})_c + g\cos\gamma\right]}{qS\,C_0\,H_5 - \cos\varphi} \qquad 5:7$$

where

$$K^* \equiv -m/T^* \qquad 5:8$$

In practice, K* may take on a different value from that above so as to provide approriate closed-loop responsiveness and stability of the guidance system.

Usually, the first command equation, i.e., on $\dot{N}_c^{(1)}$, would not be implemented, because control over $\dot{V}$ is weak when the directions the thrusters fire are always normal to the projectile x axis.

The other two values of $\dot{N}_c$ in Eqs. 5:5 - 5:6 may each be submitted to testing against respective thresholds, and the decision to fire an impulsive thruster may be made if one or more of these threshold tests is/are affirmative. However, such a policy leads to firing if the expected result along any axis is favorable in spite of any adverse repercussions along the other axes; therefore a decision based on a weighted sum of the individual sub-decisions is employed, taking

$$\dot{N}_c = \dot{N}_c^{(2)} \left|\sin \varphi\right| + \dot{N}_c^{(3)} \left|\cos \varphi\right| \qquad 5:9$$

The threshold on $\dot{N}_c$ is nominally set at zero, but may be raised to account for information uncertainties.

Combining Eqs. 2:28, 5:4, and 5:6

$$\dot{N}_c^{(2)} = \frac{K^* \left(U_{1_c} \cos \psi - U_{2_c} \sin \psi\right)}{qS\, C_0\, C_2 \cos \varphi - \left(qS\, C_0\, C_{N\alpha} + 1\right) \sin \varphi} \qquad 5:10$$

and combining Eqs. 2:30, 5:3, and 5:7

$$\dot{N}_c^{(3)} = \frac{K^* \left[U_{1_c} \sin \gamma \sin \psi + U_{2_c} \sin \gamma \cos \psi - \left(U_{c_3} + g\right) \cos \gamma\right]}{\left(qS\, C_0\, C_{N\alpha} + 1\right) \cos \varphi + qS\, C_0\, C_2 \sin \varphi} \qquad 5:11$$

Eqs. 5:1, 5:9, 5:10 and 5:11 define a baseline guidance law, diagrammed in Figure 2, that may be studied in comparison with the OPTG formulation. Note the equivalence of Eqs. 3:9 - 3:11 and 5:1 as W_5 tends to infinity. In the limit, the OPTG guidance law responds in the same way as the baseline law if total emphasis is placed on the W_5 term in the integral performance criterion.

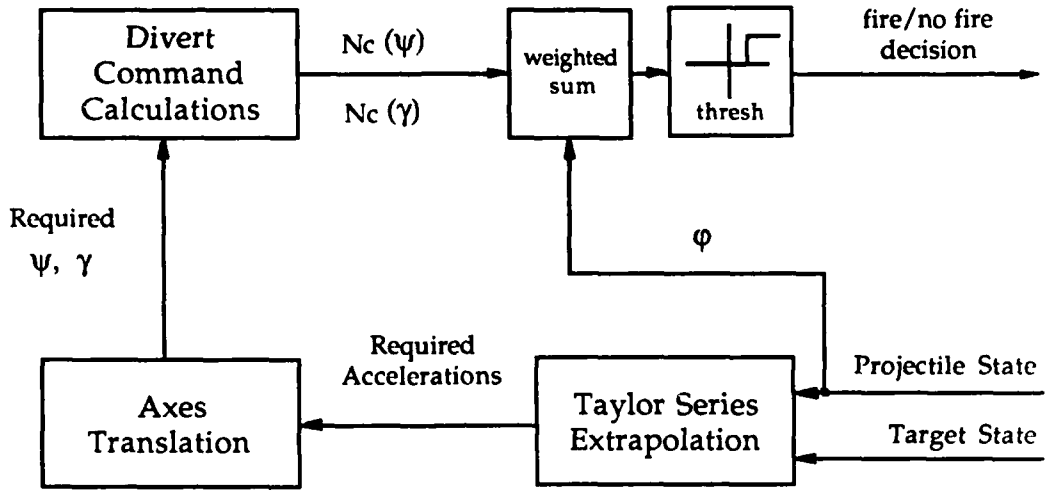


Figure 2: Baseline Guidance Computations

There are important reasons for not reducing the OPTG guidance to the baseline form. First, the OPTG guidance is a two-point boundary-value guidance law; it provides essentially zero final error if the maneuvers of the target are within allowable limits for the projectile system. On the other hand, the baseline guidance law acts to minimize the integral of squared predicted error; with the baseline formulation, the actual final error receives no more attention mathematically than does the final error predicted at each of the decision points prior to t_f. Ultimately, it is not the time history of predicted final errors but the actual error at t_f that determines the probability of target kill. OPTG guidance addresses directly the requirements on final error and, depending on the value selected for W_5, may put little or almost no weight on the time history of predicted final errors.

A second important reason for using the OPTG formulation in preference to the baseline guidance law is that OPTG guidance provides conservation of projectile maneuver resources in a way that does not compromise final error. The W_0, W_1, W_2, W_3, W_4, W_6, W_7, and W_8 terms in the OPTG formulation address the management of maneuver resources. While it is possible to incorporate a resource expenditure penalty in the baseline type of guidance law, the effect of this penalty would be to engender greater miss distances. In other words, maneuver resource management is best treated in the two-point boundary-value guidance context.

6. SIMULATION PROGRAM

The essential parts of the three-degree-of-freedom (3DOF) simulation employed are listed in Appendix E. A goal of the program is to enable one to examine flights starting from arbitrary points in space and proceeding either forward or backward in time to suitable stopping points. A system state structure is therefore defined with which one can easily store and return the required system "snapshots", or sets of variables which uniquely describe the status of the engagement under study. The states $\underline{S} = (\varphi, \underline{x}, V, \gamma, \psi, \underline{\lambda}, \underline{x}_T, \underline{V}_T, N)$ are arrived at by integrating their respective derivatives, $\dot{\underline{S}} = (P, \underline{V}, \dot{V}, \dot{\gamma}, \dot{\psi}, \dot{\underline{\lambda}}, \underline{V}_T \underline{U}_T, \dot{N})$ over time (where $\dot{N}$ is treated as either 0 or 1, as discussed in Section 4.5).

The state derivatives, $\dot{\underline{S}}$, can be written entirely as functions of $\underline{S}$; in the program, these calculations are split into common (*dyn*) and squib-related (*deriv*) parts. Note that the kinematic equations in *dyn* and *deriv* are general, the variational equations derive from the calculus presented in Section 3, and the required system-specific constants are isolated in an include file (*system.h*). The guidance command, $\dot{N}_c$, subjected to a user-defined threshold for the fire/no-fire decision, is calculated according to one of three guidance modes:

0: Ballistic ($\dot{N} = 0$)
1: Baseline (Section 5)
2: Optimum Path-to-Go (OPTG; Section 4).

The *fly* routine integrates from $\underline{S}_k$ to $\underline{S}_{k+1}$ (or vice-versa given the derivatives at the appropriate starting point) through either the rectangular method or Heun's method of integration, as selected by the user. In addition to performing a single iteration of the system, *fly* examines whether or not the timestep should be reduced or reversed to "zero-in-on" the desired final condition (such as minimum time-to-go, τ, for closest point of approach). Not shown in the listing is the capability of storing the system state and performing a forward look-ahead -- a rapid extrapolation of current

information to examine its closed-loop implications -- which is required if miss-distance sensitivity APNs are used.

The system is flown as long as allowed. The conditions allowing flight vary with flight direction according to the routine *conds_allow*. Also depending on direction, the system is initialized or finalized in *main* after the state and control variables have been retrieved from the user interface and translated into internal units. The interface (not listed) is a user-friendly general package developed by Barron Associates to improve flexibility and portability in simulation development. With it, the user may save or restore named sets of variables, change as many or as few parameters as desired between trials, automatically reset values to defaults, and get descriptive information on the purpose of each program option.

When the user quits a series of simulated flights, or trials, in one direction (*more_trials*), the option to examine flights in the other direction is presented. In addition to end-point condition manipulation through the interface, the user may define a set of parameters to adjust, and initiate, certain automatic explorations according to the following search modes found in *advise*:

0: None: use interface
1: Grid: search in a regular pattern from loops within loops.
2: Grope: employ the Guided Random Optimizer of Performance Error (GROPE), a search algorithm developed by Barron Associates (not listed). GROPE attempts to minimize a criterion defined by the user in *main*, (for ex: miss distance). The search algorithm is capable of efficient exploration of high-dimensioned multi-modal surfaces.
3: Gauss: randomly select values for parameters according to a zero-mean gaussian distribution of user-defined standard deviation.
4: Uniform: employ a uniform distribution bounded in each parameter by the user.
5: "Firtree": use a special sensitivity-APN database generation mode.

The bounds to employ when changing the parameters are defined in the include file *bounds.h*.

Also present in the program are options to select a wide or narrow screen of published information separated by adjustable time intervals, output a history of state variables for presentation or analysis, enter a "debug" mode to receive interior details on certain calculations, define the record modes to employ for efficient database generation, and to "bounce", or return in the opposite direction, according to the same or a different guidance mode. (For instance, one could simulate a reverse trajectory under OPTG guidance and "bounce" forward ballistically to compare the divert, or difference in final position.) Not shown is the use of adaptively-synthesized polynomial networks (APNs) for guidance initialization and/or adjustment in the closed-loop OPTG system.

7. OPTG AND BASELINE GUIDANCE SIMULATION RESULTS

Use of the 3DOF CAT projectile simulation software presented in Section 6 shows the OPTG guidance law consistently producing smaller terminal miss distances, using fewer squib firings, and providing greater maximum divert capability than does the baseline guidance. Section 1 provides some discussion of these effects; here we present detailed results supporting the conclusions reached.

7.1 Engagements Simulated

The engagement geometry simulated is that of a maneuvering or non-maneuvering target located in the x_1-x_3 plane at a distance x_2 = 10,000 ft. from the gun muzzle (where x_1, x_2, x_3 correspond to "east", "north", height position). Target accelerations are tracked with some degree of uncertainty (gaussianly-distributed random measurement noise) and are, in the simulations studied, either constant or governed by the equations:

$$u_{T_1}(t) = g \sin(\omega t)/3 \quad \text{ft./sec.}^2 \qquad 7:1$$

$$u_{T_3}(t) = -32.8\, \omega^2 \sin(\omega t) \quad \text{ft./sec.}^2 \qquad 7:2$$

where ω is 0.1π rad./sec. The equations are derived from the elliptic target motion relationships defined in Reference 4:

$$y(t) = g\,(t - \sin(\omega t + \varphi_y)/\omega)/3\omega \quad \text{meters} \qquad 7:3$$

$$z(t) = 50 + 10 \sin(\omega t + \varphi_z) \quad \text{meters} \qquad 7:4$$

where (y, z) is (x_1, x_3), and the random target phase components, φ_y and φ_z, are omitted.

Initial projectile conditions were constant: angle of inclination, $\gamma = 2°$; muzzle velocity, V_m = 3821.5 ft./sec.; roll attitude, $\varphi = 0$; muzzle attitude east of north, $\psi = 0$; (constant) roll rate, P = 21 Hz.; and decision time step, Δt = 0.05

sec. (Note that the latter two parameters cause an effective decision roll rate of 1 Hz.) An unguided projectile (i.e., ballistic trajectory) was found to take approximately 3.065 seconds, and to cross the 10,000 ft. downrange plane at an x_1 ("east") position of 0.0 ft., and an x_3 (height) position of 182.23 ft. -- a location henceforth referred to as the "ballistic point".

7.2 Simplified OPTG System

The complete OPTG guidance system envisioned (diagrammed at a high level in Figure 1.a) is detailed in Figure 3.a. Adjoint differential equations are integrated, and the squib fire command calculated, at each decision point to determine whether a squib will fire, diverting or not diverting the projectile at that point in its trajectory. Projectile and target state information is filtered (potentially using linear polynomial filters, a class of estimation and prediction filters) to track progress and update the OPTG guidance equations. Those equations are initialized and, if necessary, periodically re-initialized, by an APN synthesized off-line from simulations of representative trajectories. The APN of Figure 3.a relies on a preceding explicit prediction of the intercept point; but, as shown earlier in Figure 1.b, this task may be subsumed by a more comprehensive APN which works directly from the filtered target and projectile state measurements.

The simplified OPTG guidance system prototyped and evaluated in this first phase of work is shown in Figure 3.b. As the class of engagements studied terminates at a nearly constant range, intercept point prediction is based on the ballistic flight time. The target position estimator, using this τ and the true initial target position, was simulated to be only a minor source of noise in the experiments. (A complete examination of this important topic -- beyond the scope of this study -- will be required for full Optimum Path-To-Go guidance implementation). Note that, unlike full OPTG guidance, the simplified version performs no guidance re-initialization or adjustment; it is the "open loop" component of the eventual system.

In the cases examined, the constant OPTG weights defining the basic character of the trajectories, W_0 through W_8, were, in order: { 10, 0, 0, 0, 100, 0.001, 0, 0, 0 }. Note that the adjoint equation weight which employs target

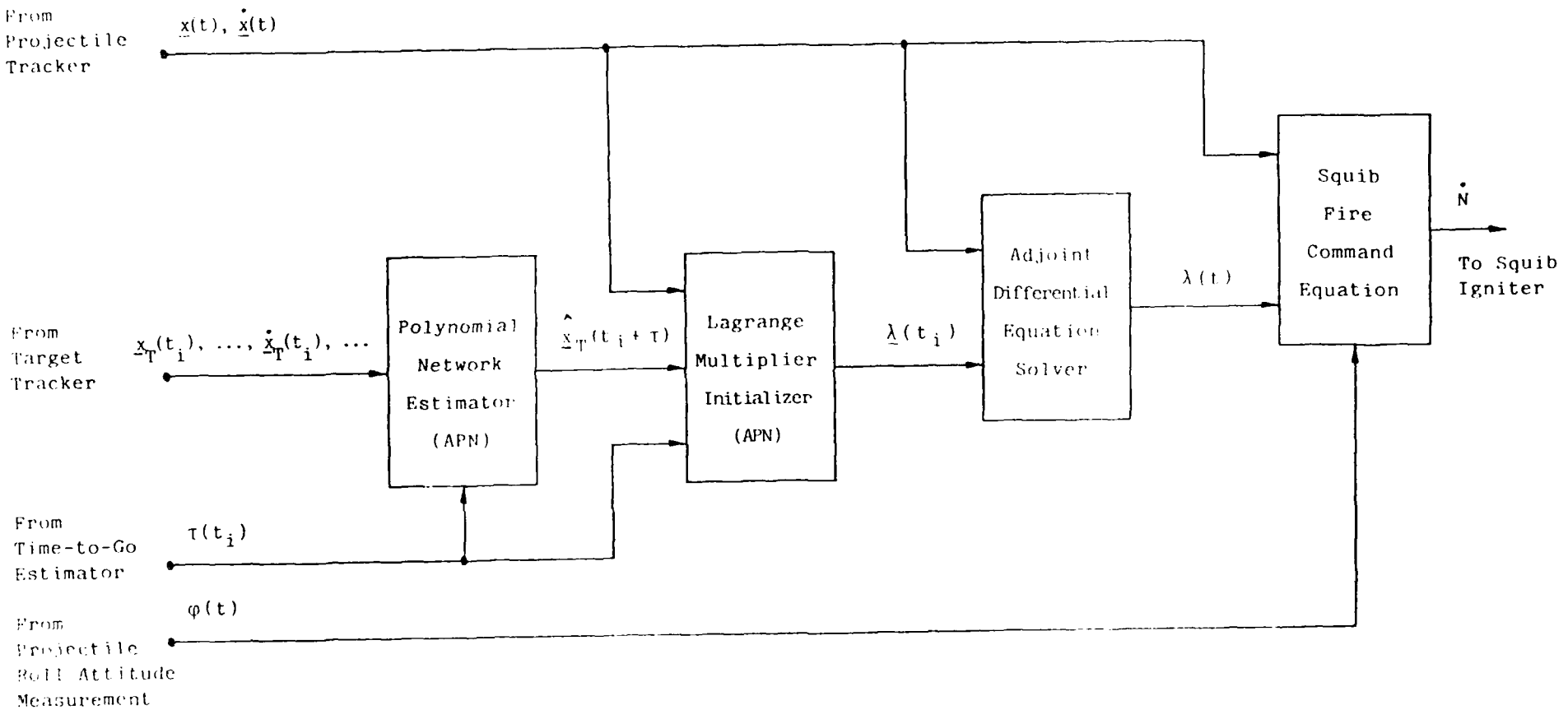


Figure 3.a: Detailed Optimum-Path-to-Go Guidance Law

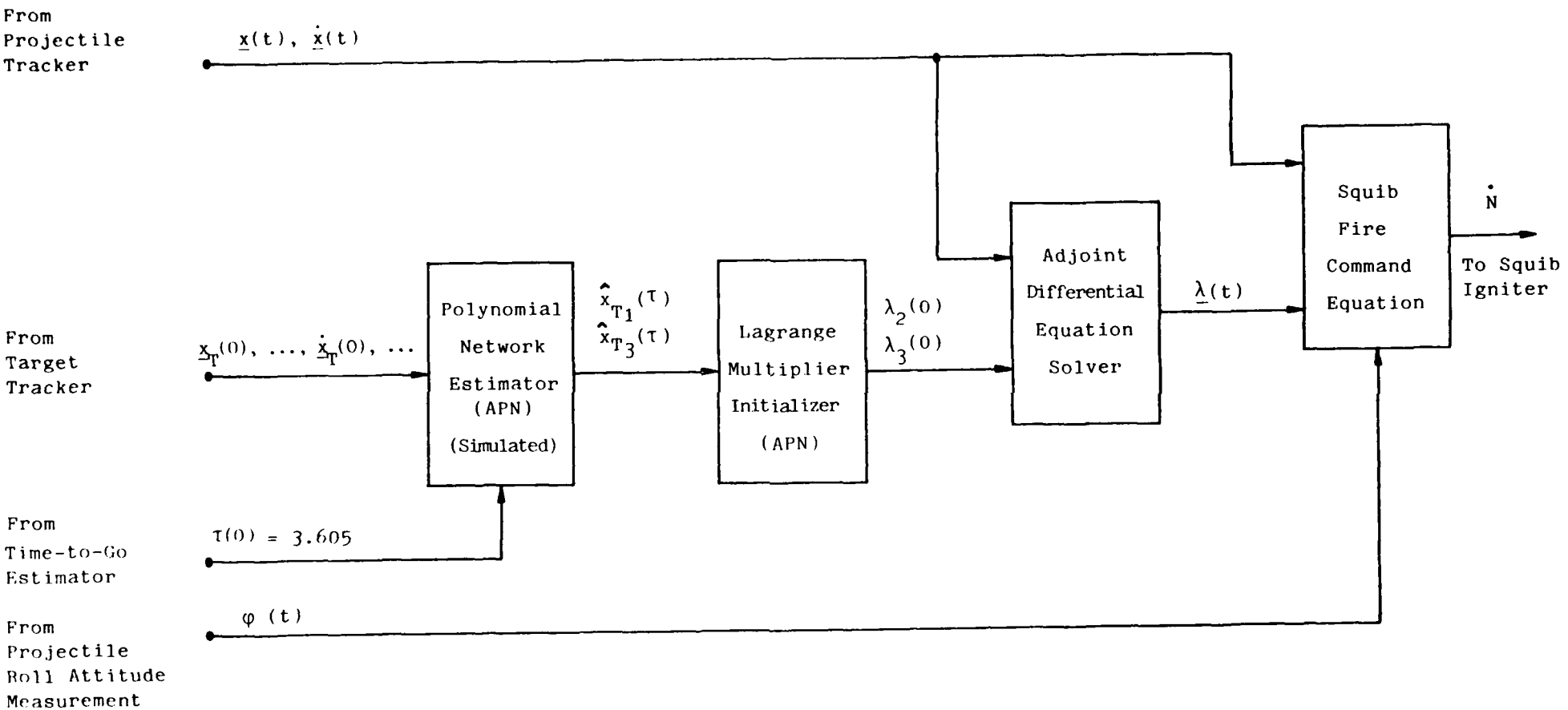


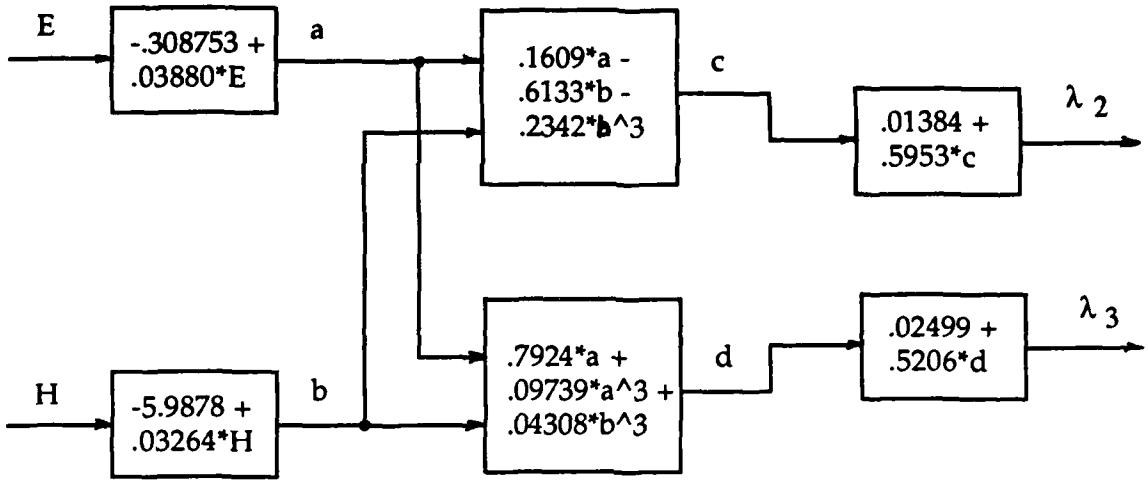
Figure 3.b: Prototype Optimum-Path-to-Go Guidance Law Simulated

state feedback (W_5) was disabled; in fact, only the $\dot{N}_c$ scaling weight, W_0, and the squib conservation weight, W_4, were used to govern the basic OPTG tradeoffs. (The feedback weight, W_5, though shown with a small value, is effectively disabled when the kinetic energy weights, W_1,W_2,W_3, are absent.) Although most of the weights (such as W_6, W_7, W_8, the downrange, crossrange, and altitude change weights) were studied, it was found that the small subset above was sufficient to demonstrate the necessary concepts. However, further research could likely uncover more trajectory enhancements by examining the full use of these temporarily disabled weights.

7.3 OPTG vs. Baseline Guidance

To examine the operational range, or "handprint", of the optimum-path-to-go strategy, initial values of two of the controlling Lagrange multipliers, λ_2 and λ_3, were varied randomly in simulation between values of +/- 1.50, with all the other λ's starting at 0.0. (Note that OPTG guidance reduces to the ballistic case when the entire vector, $\underline{\lambda}$, is zeroed.) The database of engagements thus generated was filtered to reduce redundant impacts, retaining trajectories firing the fewest squibs in the case of a conflict. The Algorithm for Synthesis of Polynomial Networks (ASPN) was then employed to synthesize a model (an APN) capable of revealing the appropriate $\underline{\lambda}_0$ values to use when given the expected final target position ($x_{T_{1_f}}$ and $x_{T_{3_f}}$). Using "E" and "H" for expected position, Figure 3.c details the resulting initializing APN, which network consists of a pair of "carved double" elements, surrounded by "normalization" and "unitization" nodes (Ref. 12).

When installed in the simulation, and relied on to initialize the OPTG guidance for a host of target position cases within the maneuver envelope, the APN proved to be very successful. Against motionless targets, the OPTG handprint in the miss distance plane, shown in Figure 4.a, is quite circular, providing good controllability over the entire region physically reachable by the projectile. Examination of the data reveals, importantly, that within the performance boundaries, the accuracy of the OPTG system does not degrade substantially with required divert, or final distance from the ballistic impact



Approximate Equations:

lambda_2 = 32.10 - 0.5014*H + 0.3717e-2*E +
0.2668e-2*H^2 - 0.4849e-3*H^3

lambda_3 = -4.919 + 0.07874*H + 0.01657*E -
0.4292e-3*H^2 - 0.7073e-4*E^2 + 0.2963e-5*E^3
+ 0.7799e-6*H^3

Figure 3.c: Prototype Initializing APN

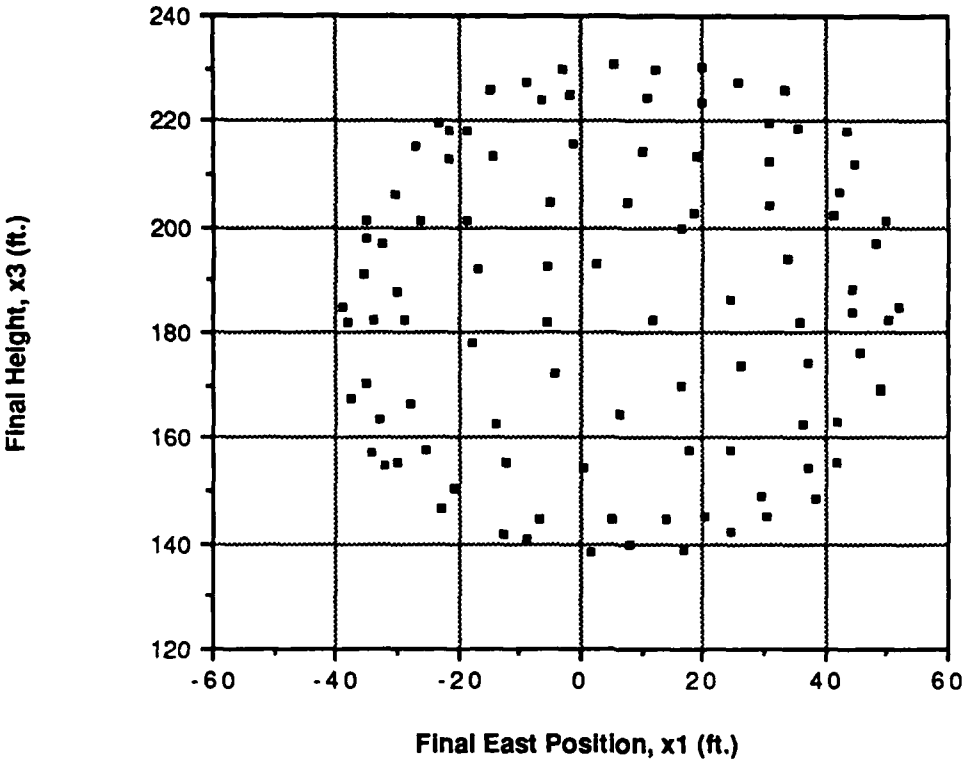


Figure 4.a: OPTG Guidance Handprint (0 Threshold)

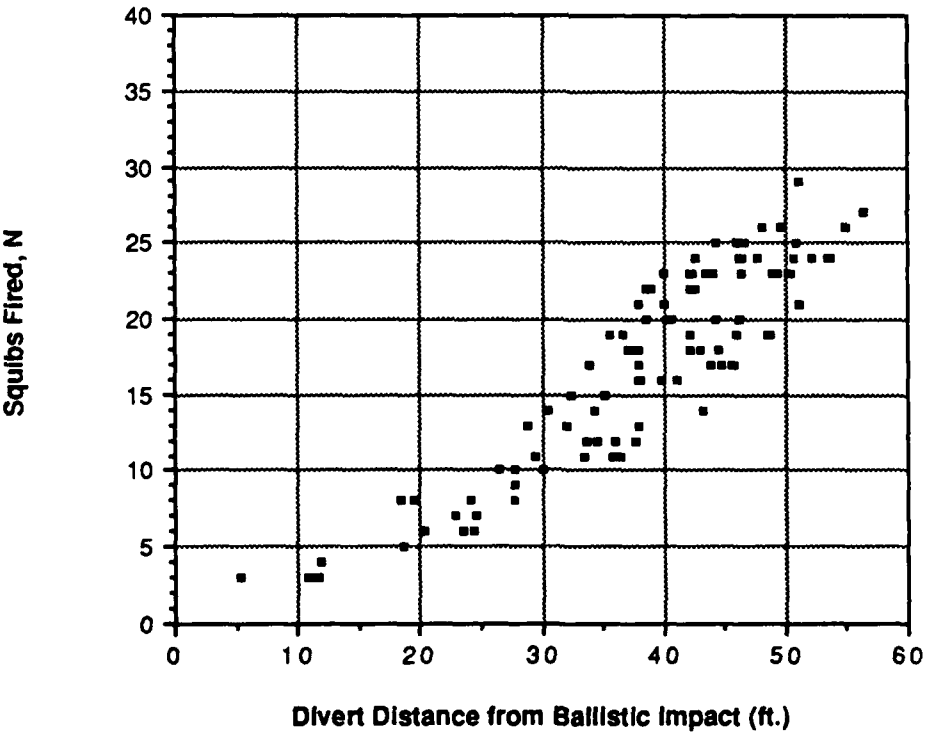


Figure 4.b: OPTG Guidance Squib Usage (0 Threshold)

point. Figure 4.b reveals that there is a clear relationship between squibs fired and the resulting divert; in fact, the number of squibs needed in a particular engagement is about one half the total divert (in feet) required.

When the baseline guidance law is sent after the same set of targets, the handprint depicted in Figure 5.a results. Its pattern is more that of a diamond than a circle; significantly, points at the corners of the OPTG maneuver boundary appear to be unreachable with the baseline method. Many more squibs are being fired, as shown in Figure 5.b, but no clear relationship exists between thruster energy expended and the resulting divert (a consequence partly of the unbridled continuous correcting performed by the baseline law).

Note that if the projectiles in the above baseline case were limited to 40 squibs, most trajectories would have had many fewer opportunities to refine final accuracy, with a consequent (but slight) worsening of miss distance. The difference is only slight because, for the squibs to divert the projectile significantly, they must be fired early enough for the effects to integrate throughout the flight. The loss of final refinements when resources are exhausted is therefore a consideration less important than the proper early use of those resources. Both issues are successfully addressed by the OPTG guidance method, which inherently conserves resources by acting early -- a property not shared by late-acting pursuit, proportional navigation, or predictive proportional navigation (e.g., the baseline) guidance laws.

Putting a threshold on the guidance command equation -- essential when uncertainties in target accelerations exist -- reduces the squib firing called for by the baseline guidance law. Figures 6.a,b depict the results when a threshold of 0.2 is employed in the baseline law. Now a definite relationship exists between squibs fired and divert achieved (6.b), and though the system uses fewer squibs than without a threshold, it still fires roughly 8 more squibs than with OPTG guidance. The price of saving squibs through a threshold, however, is revealed in the resulting miss distance, as suggested by the distorted and shrunken handprint (6.a). The accuracy of the conventional guidance law in this, the simplest of engagement scenarios, is good below a divert of about 32 feet, as revealed in Figure 7.a, but worsens quite sharply

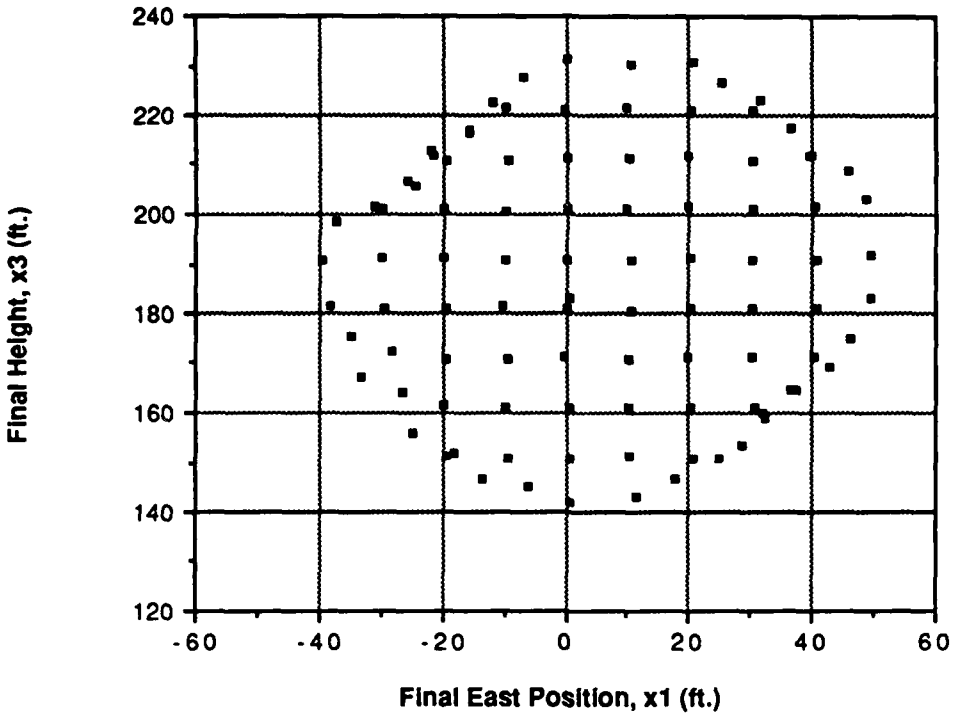


Figure 5.a: Baseline Guidance Handprint (0 Threshold)

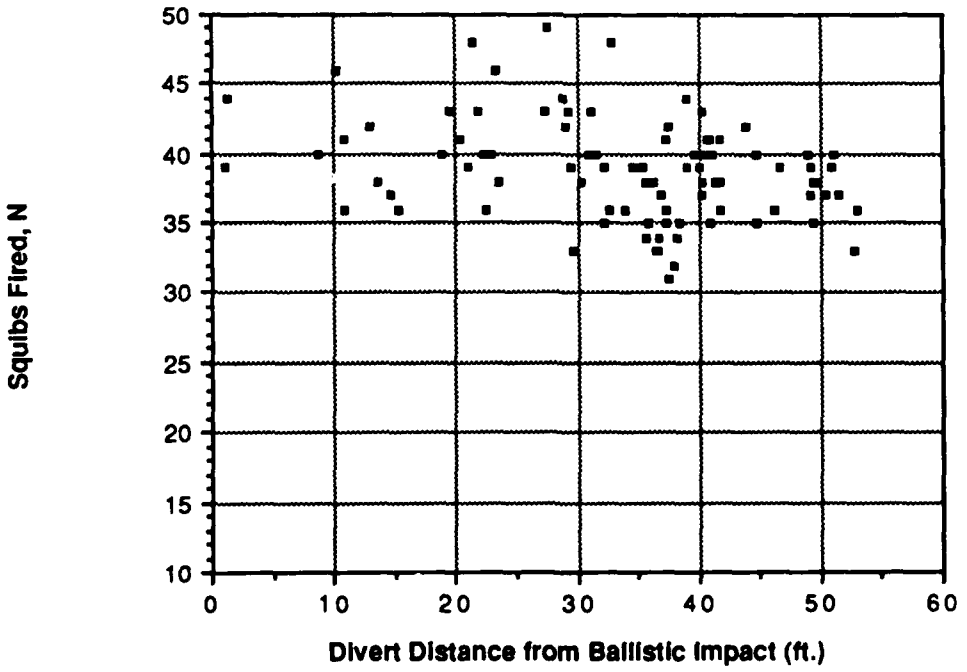


Figure 5.b: Baseline Guidance Squib Usage (0 Threshold)

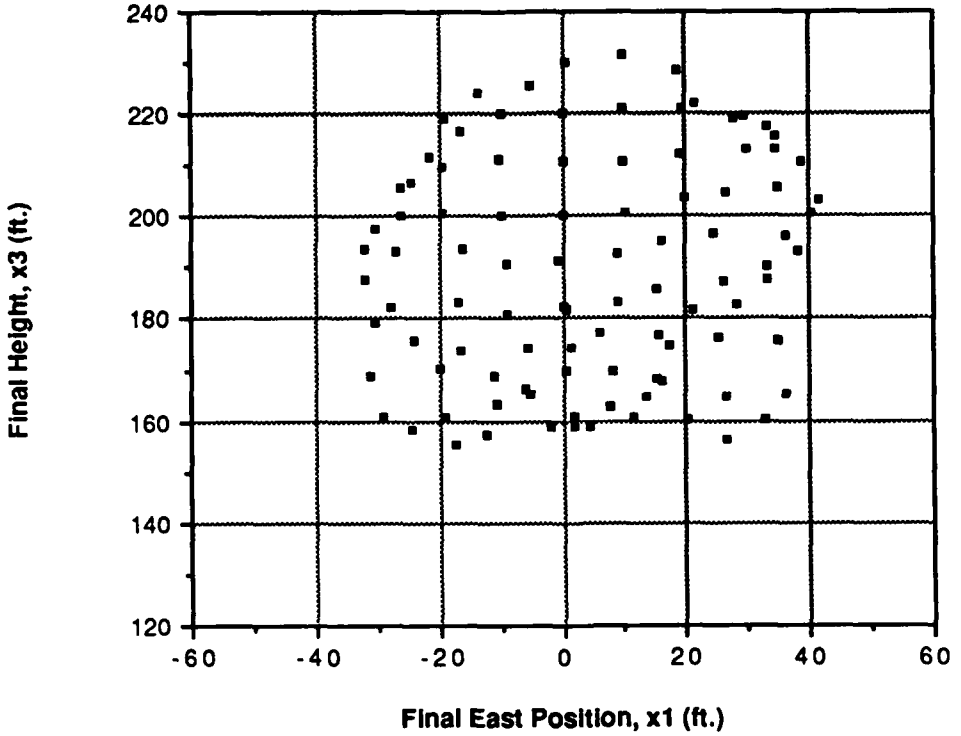


Figure 6.a: Baseline Guidance Handprint (0.2 Threshold)

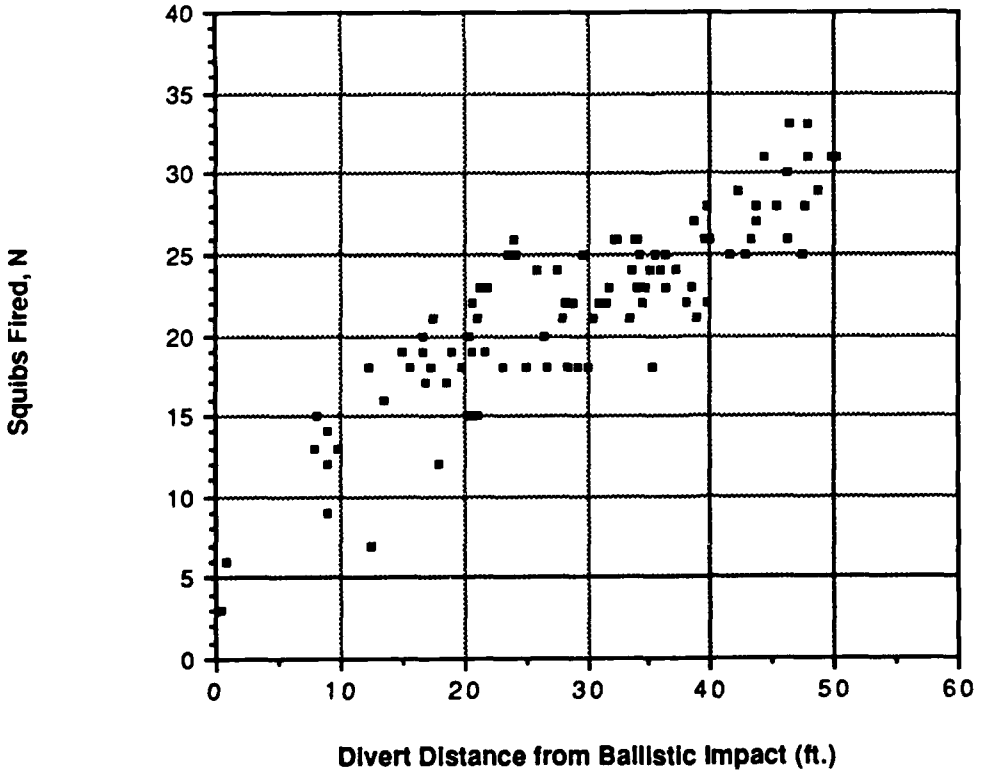


Figure 6.b: Baseline Guidance Squib Usage (0.2 Threshold)

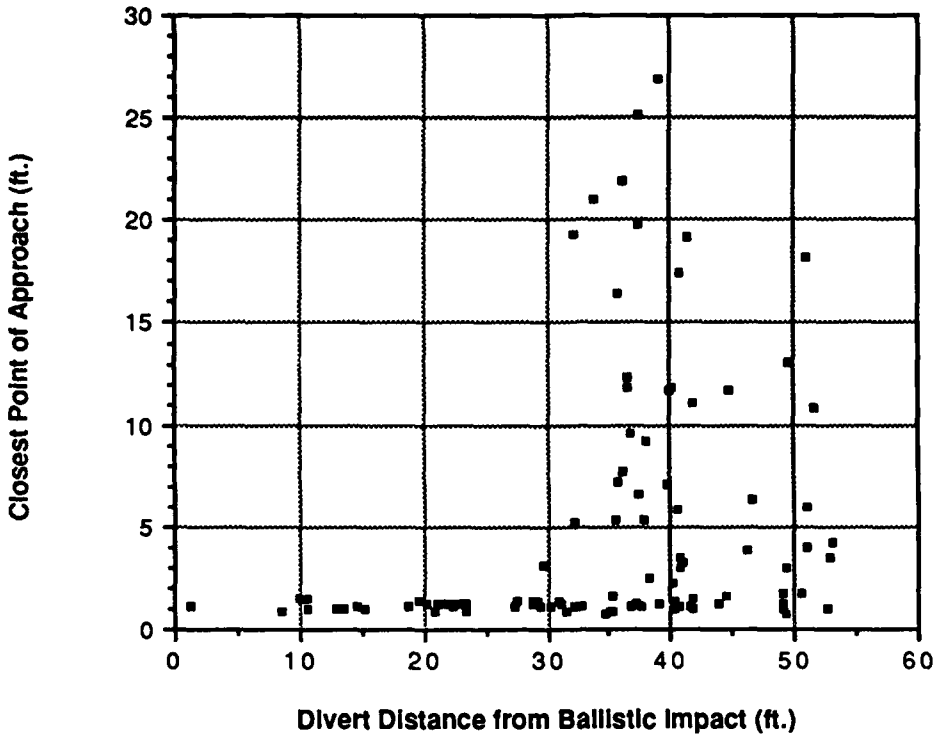


Figure 7.a: Baseline Guidance Performance (0 Threshold)

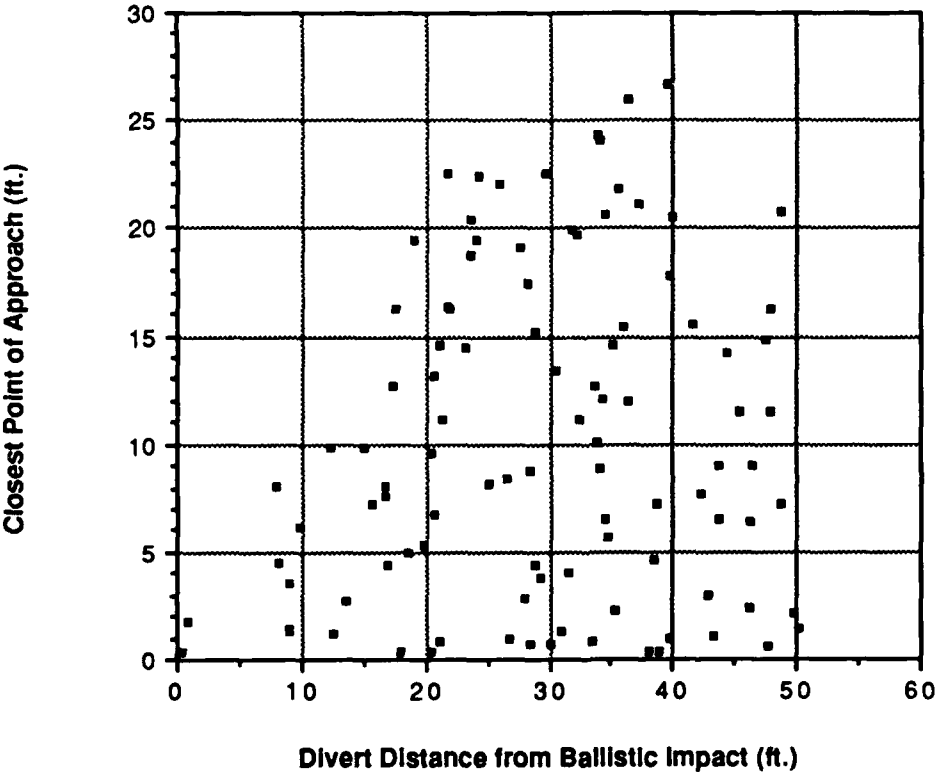


Figure 7.b: Baseline Guidance Performance (0.2 Threshold)

beyond that. When the thresholds necessary for use in noisy environments are employed, the accuracy of the baseline guidance only worsens (Figure 7.b).

The OPTG guidance method is only slightly affected by use of the same guidance command equation threshold (0.2). Though the APNs were not trained on such a threshold, its presence served mostly to delay the onset of each fire decision, resulting in slight "twists" of the intended trajectories. The end effect was minor, and the trajectories have a very similar miss distance distribution, confirming that OPTG laws can be made much less sensitive to the values of any thresholds employed to deal with uncertain (noisy) information. Figure 8 reveals that, with no squib fire decision threshold, the OPTG trajectories achieved given divert distances with about 10 fewer squibs than the thresholded (i.e., most competitive) baseline method.

Note that, as initialization and re-initialization of an optimum-path-to-go guidance law occurs only a few times throughout a flight (and can be aperiodic and data-driven), noisy input data may be filtered for a relatively long time before use, reducing the effect of tracking noise. Also, as the OPTG technique is of the two-point boundary-value class, an intercept goal is always before the projectile. Thus, data drop-outs or loss of link with host system ("maximum noise") can be dealt with in a straightforward manner: stay on course to the last predicted intercept point.

7.4 Blended OPTG/Baseline Guidance

To compare the guidance techniques in closed-loop (say, against maneuvering targets), it is necessary to modify the prototype OPTG guidance by switching to its competitor (the baseline method) in the end-game (as entirely OPTG closed-loop guidance synthesis is beyond the scope of this Phase I feasibility study). For this reason, the closed-loop comparison results, especially in squib conservation, are slightly less dramatic than would be possible with a pure OPTG method (such as that proposed for Phase II) which employs target feedback information; nevertheless, the improvements are significant.

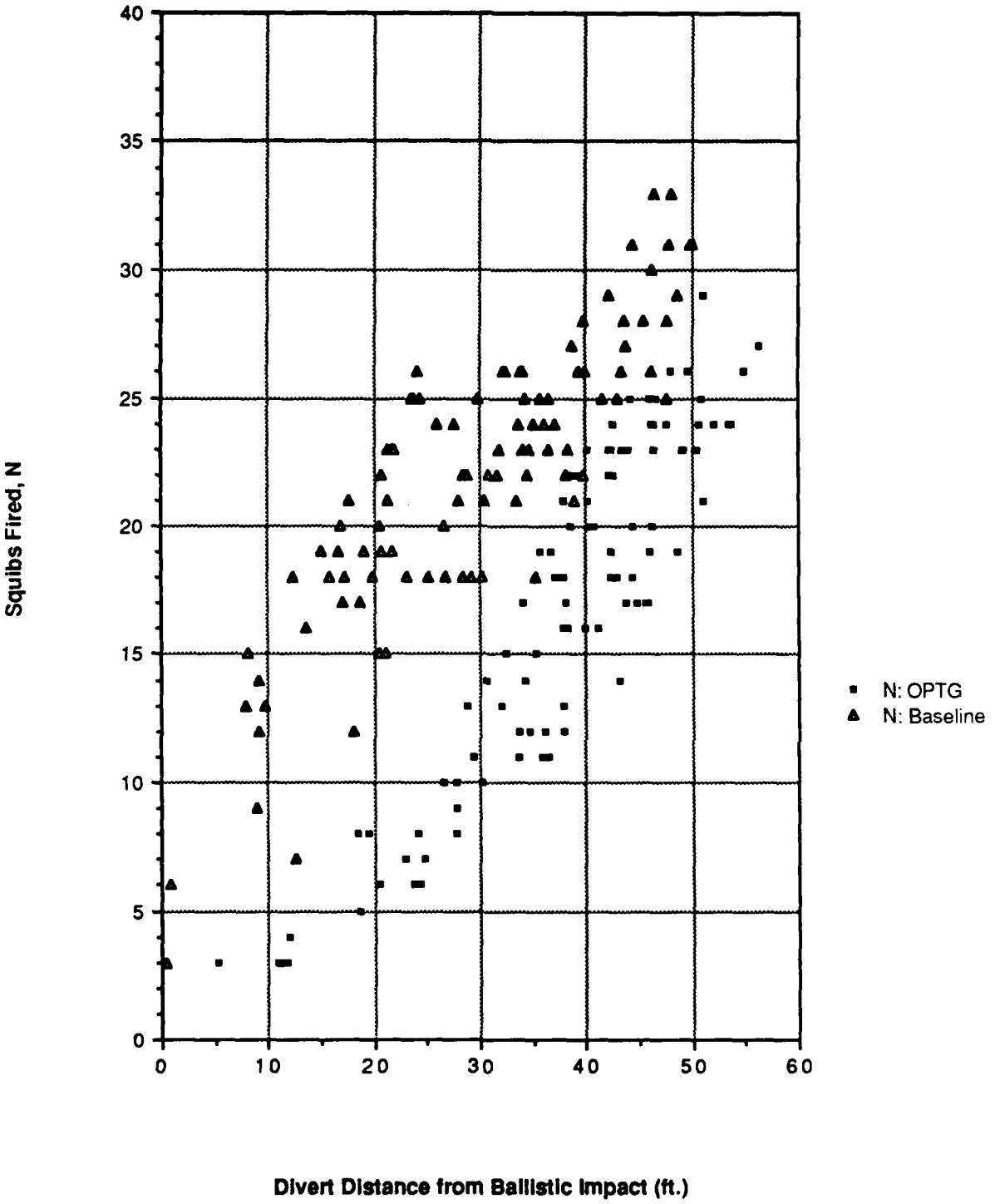


Figure 8: Comparison of Squibs Fired vs. Divert Distance

In this "blended" guidance law, the OPTG system of Figure 3.b becomes the initializing component, and the baseline law (of Figure 2) is used in the end-game. Switching to the baseline guidance method in the final second of flight provides a workable, if sub-optimal, way to follow the target for preliminary evaluation purposes. The performance of a complete OPTG system can clearly be lower-bounded by this blended version; still, as shown below, even it outperforms what is a strong conventional technique: the pure baseline guidance method.

The blended method takes advantage of the early OPTG guidance decisions (when they can have the greatest effect) and the closed-loop accuracy of the baseline method (which is good when the required divert is small). When firing at stationary, but noisy targets switching to baseline guidance from OPTG guidance when the estimated time-to-go, Tau, is below 1.0, provides the accuracy results summarized in Figure 9 (and analyzed below). In short, the accuracy of the blended system is slightly better than OPTG and much better than any baseline case. As expected, the average number of squibs fired is intermediate to that of each of the techniques when employed alone -- the squib use distributions of which are depicted in Figure 10. Use of a threshold with the blended method was found to affect these results only very slightly.

7.5 OPTG Guidance Accuracy Potential

As the firing of a squib is a discrete event, the projectile handprint only approximates a continuum; actually, a given trajectory contains N decision points separated by Δt seconds, and therefore a maximum of 2^N possible positions at any given range.* "Perfect" guidance could thus be defined as the ability to reach all such possible final positions. Significantly, in both theory and practice, only OPTG guidance can approach this limit.

While the accuracy of conventional guidance techniques degrades in rough proportion to the divert (distance from the ballistic point) required, it

* For the given $x2_f$ and Δt, N is relatively large (greater than 60) so the continuum approximation is appropriate.

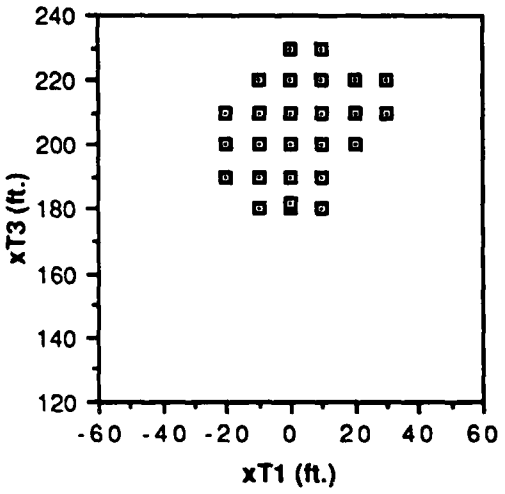


Figure 9.a: Baseline Targets with CPA < 3 ft.

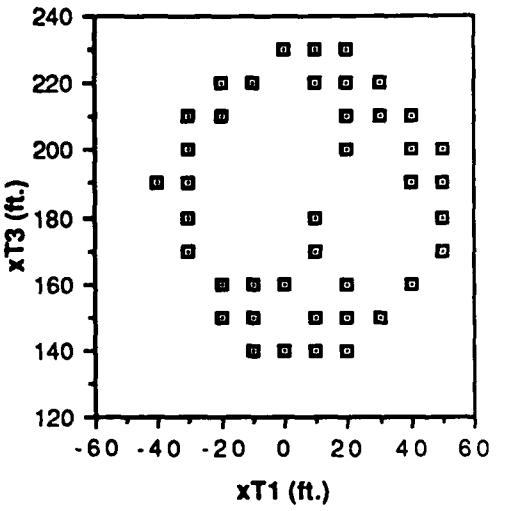


Figure 9.b: Blended Targets with CPA < 3 ft.

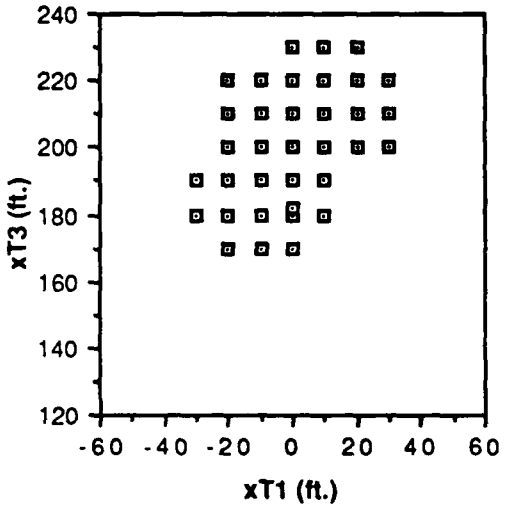


Figure 9.c: Baseline Targets with CPA < 6 ft.

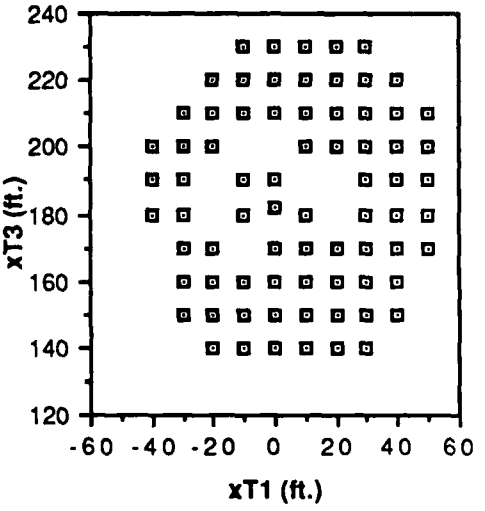


Figure 9.d: Blended Targets with CPA < 6 ft.

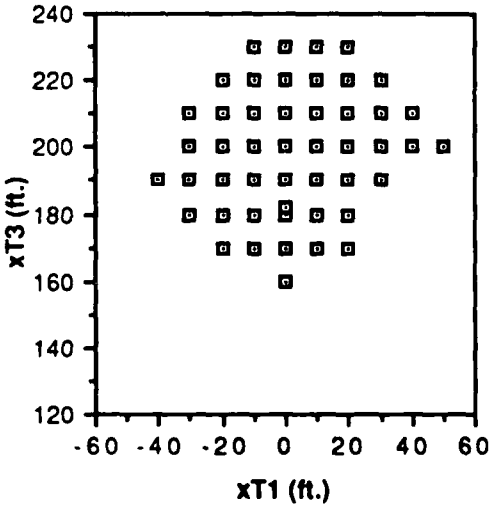


Figure 9.e: Baseline Targets with CPA < 9 ft.

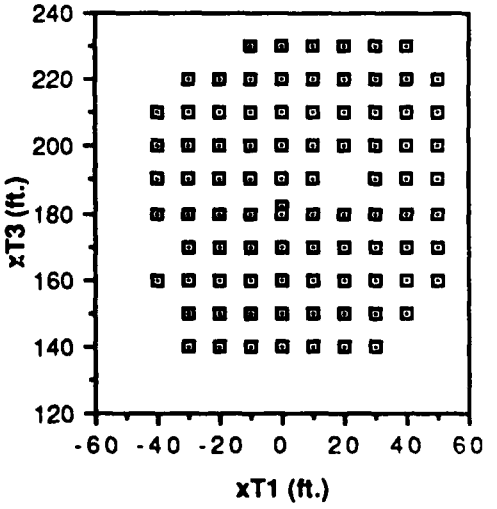


Figure 9.f: Blended Targets with CPA < 9 ft.

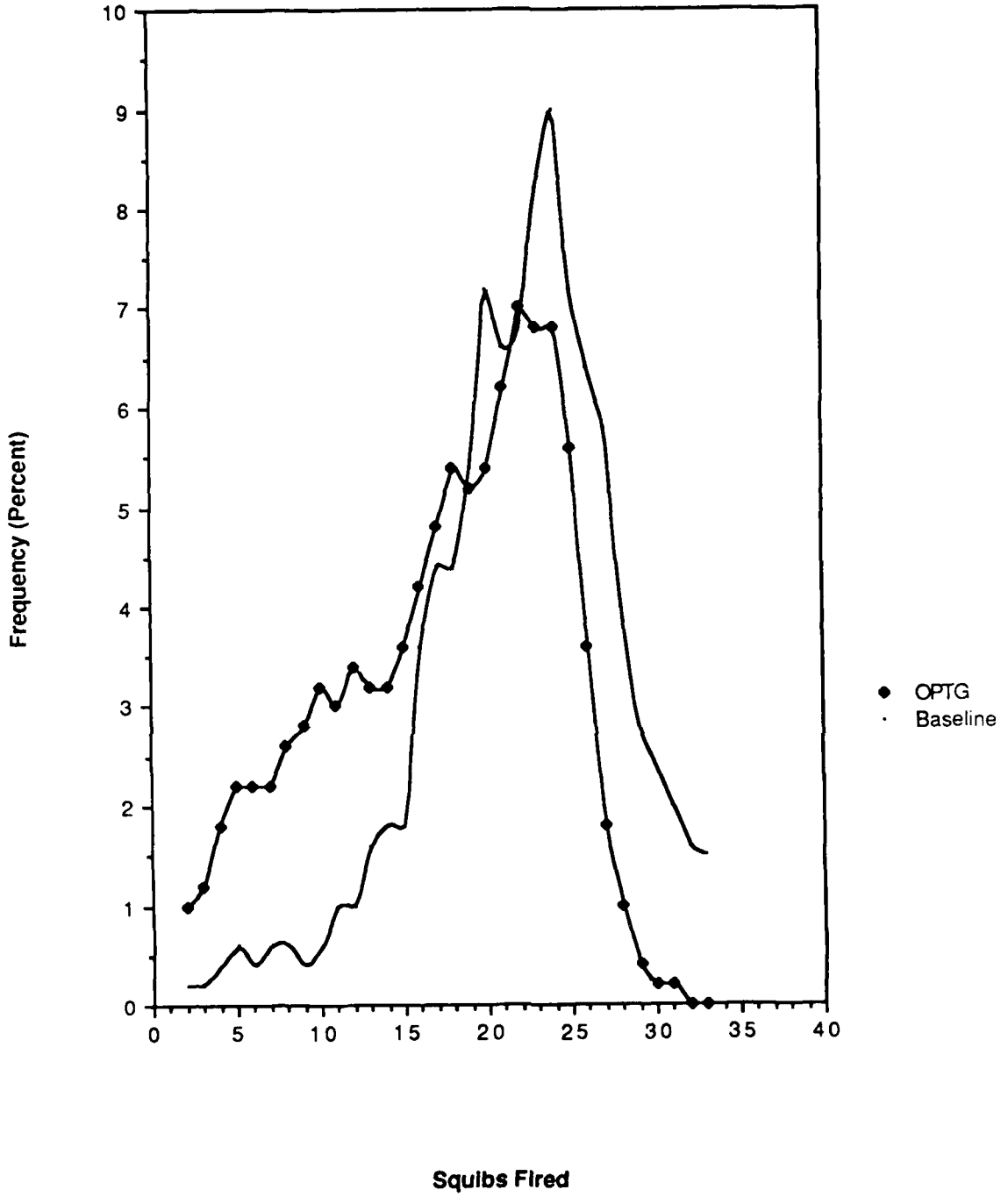


Figure 10: Squib Usage Distributions

is possible for an OPTG-guided projectile to be accurate throughout the entire envelope of positions reachable by the projectile; i.e., to approach perfect guidance. This capability is foreshadowed, or hinted at, by use of the simple initializing APN -- as can be seen from Figure 9, which compares baseline and blended guidance handprint subsets for various accuracy thresholds. For both techniques, more and more targets can be defined as "hit" as the threshold on the miss distance allowed rises from 3 to 6 to 9 feet. With the baseline method, the handprints expand outward from a "tilted" region above the ballistic point, into a diamond shape -- demonstrating that, for conventional methods, guidance accuracy and the size of the maneuver envelope are inextricably linked.

With the blended (OPTG/baseline) method, however, virtually the entire maneuver envelope is represented at each level of miss distance (or closest point of approach). The handprint fills out and expands slightly as the miss distance threshold rises, but reveals the basic circular shape of the envelope of potential maneuvers from the beginning. As is common with APNs, the best points (the target locations for which the model is most accurate) are those best represented in the database; here, a ring with a roughly 20 ft. divert radius centered around the ballistic point. Synthesis of a more accurate and complete initializing APN could easily fill out and round out this ring, leading to a 3-ft.-miss OPTG envelope covering about 6400 $ft.^2$ -- almost four times larger than the 1700 $ft.^2$ area of the comparable baseline handprint.

Refinement of the initializing APN and, more importantly, synthesis of re-initializing APNs (which would allow the method to be purely OPTG), are sure to improve the handprints, and the authors are confident that optimization of the initialization process is achievable and will lead to system performance, in the absence of other effects, approaching that of perfect guidance for a reasonable definition of acceptable miss distance.

Figure 9 compares the "hit distribution" of both guidance methods for a sample of 101 low-acceleration-noise engagements (mean and standard deviation = -0.5 $ft./sec.^2$), and Figure 11 presents that data in the form of the cumulative number of targets hit for a given allowable closest point of

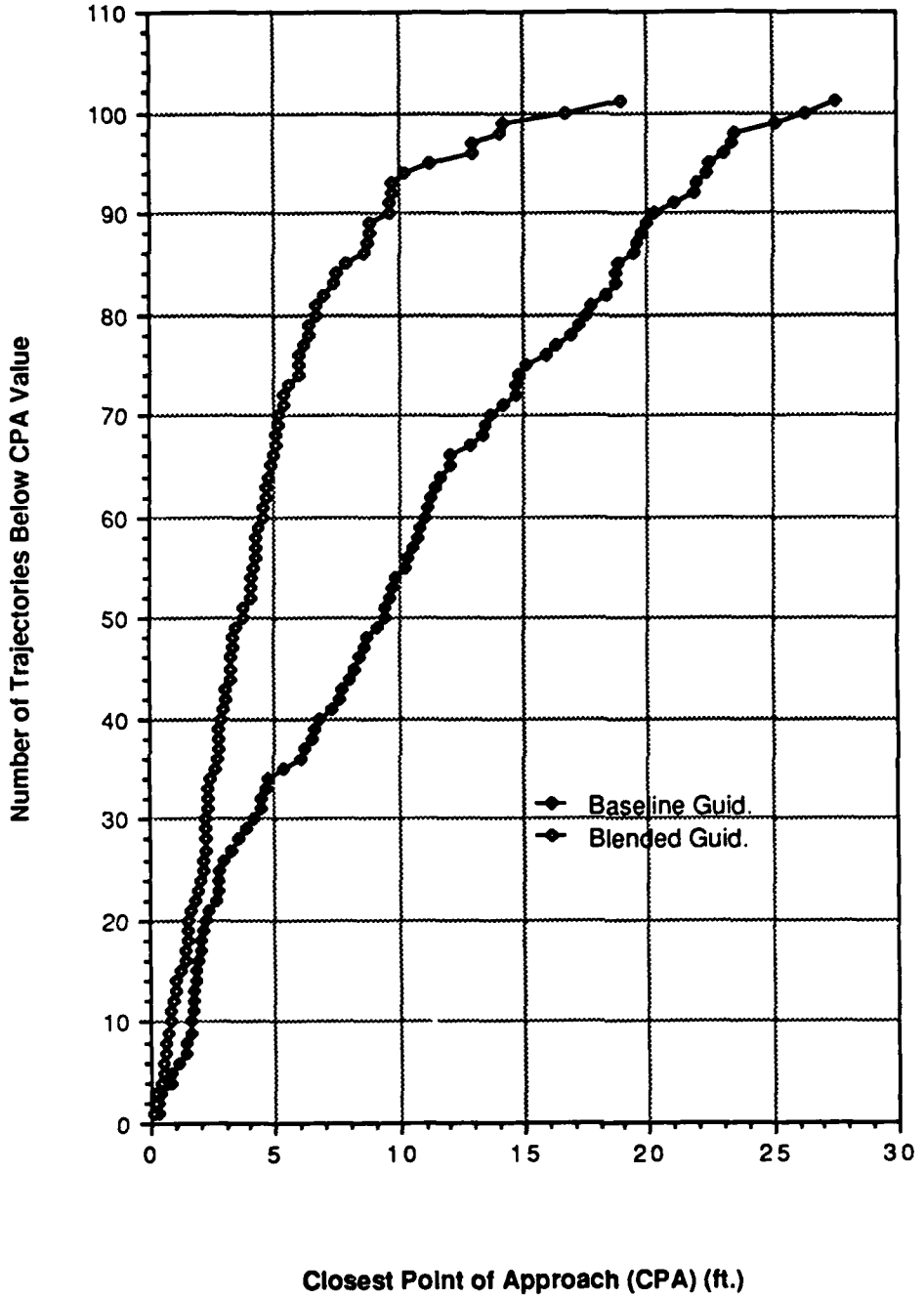


Figure 11: Cumulative CPA Comparison

approach (CPA) (i.e., the integral of the histogram, or density distribution vs. CPA). Note that at every value of CPA the blended guidance outstrips the baseline technique; only a third of the baseline trajectories miss by less than five feet, for instance, while two thirds of the blended trajectories do. The median CPA of the baseline engagements is 10 ft., compared to the blended median of 4 ft. If a CPA of 10 ft. is allowable, over 90% of the blended engagements are hits. As the following section reveals, such improvements are possible for maneuvering targets as well.

7.6 Maneuvering Target Results

Against targets maneuvering according to Eqs. 7:1 - 7:2, with some noise on the measured accelerations (mean = -.25, σ = -.50 ft./sec.^{2}), the blended and baseline systems performed according to the handprints in Figure 12. (Note that, for comparison purposes, the bounds of the handprint region were made similar to those of prior cases by offsetting the initial target x_1 and x_3 positions by 25 and 30 feet respectively.) The OPTG-based (blended) form remains quite circular (and is closest to matching the targets, which end up in a square grid, 90 feet on a side), but the baseline handprint is distorted further from the diamond shape which it can at best attain. The average divert of the blended method was 23 percent greater than that of the baseline method (37.18 compared to 30.26 ft.), and the average CPA was cut in half (baseline CPA mean: 10.62 ft., blended: 5.26 ft.; baseline CPA standard deviation: 8.01 ft./sec.^{2}, blended: 4.86 ft./sec.^{2}).

Figure 13 depicts the handprint subsets for which the CPA was less than or equal to six feet. The circular area covered by the blended handprint (even with interior gaps) is roughly 3.5 times larger than the smaller, rectangular region of the baseline technique, indicating that OPTG guidance can expand the region of weapon effectiveness for a required closest point of approach. The CPA distribution for all of the experiments is shown in Figure 14. A simplification of the graphic information, using the statistics above, would be to summarize that the blended CPA is evenly distributed, at a height of ten percent, over a ten foot region, whereas the baseline distribution, at a height of five percent, goes out to twenty feet.

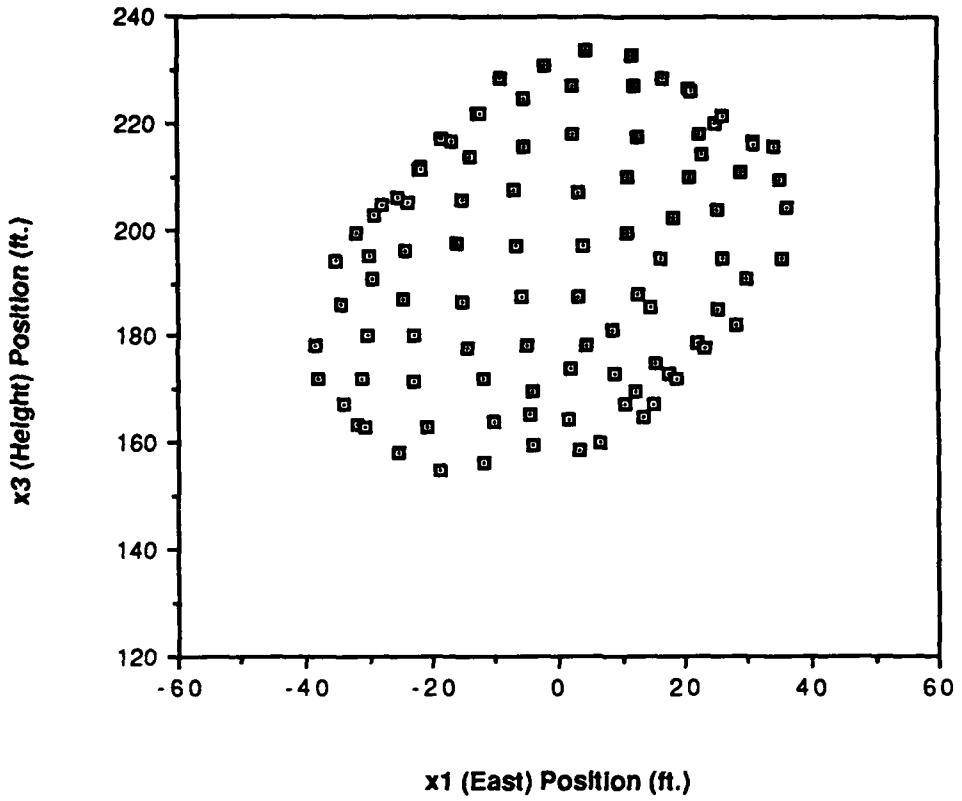


Figure 12.a: Baseline Handprint, Maneuvering Target

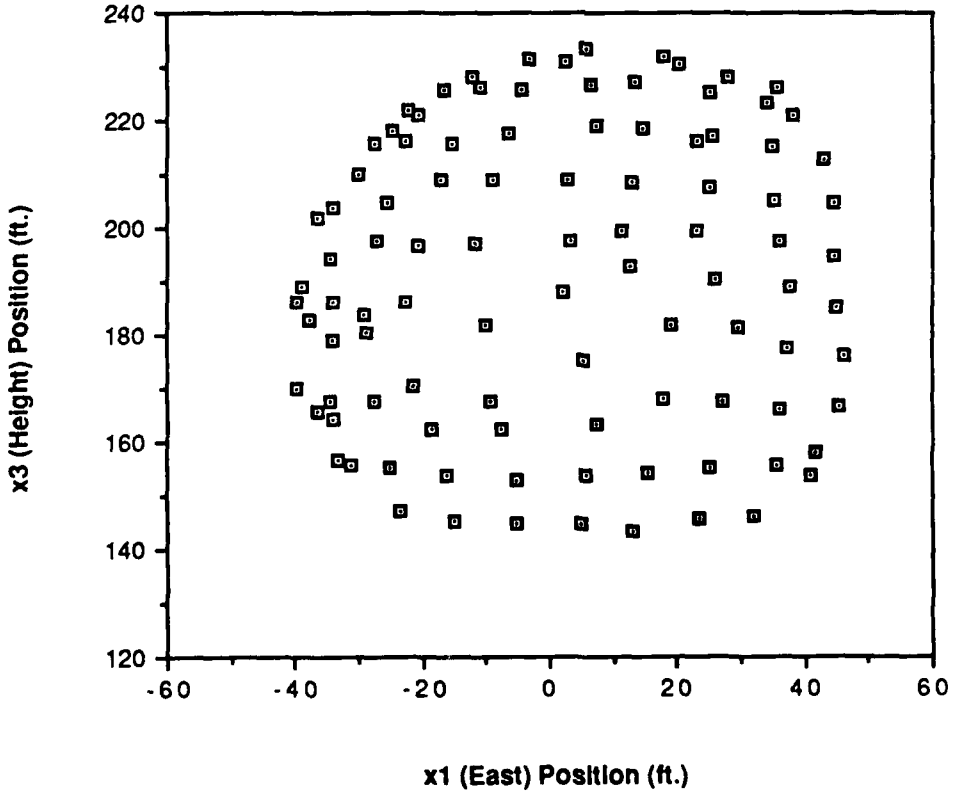


Figure 12.b: Blended Handprint, Maneuvering Target

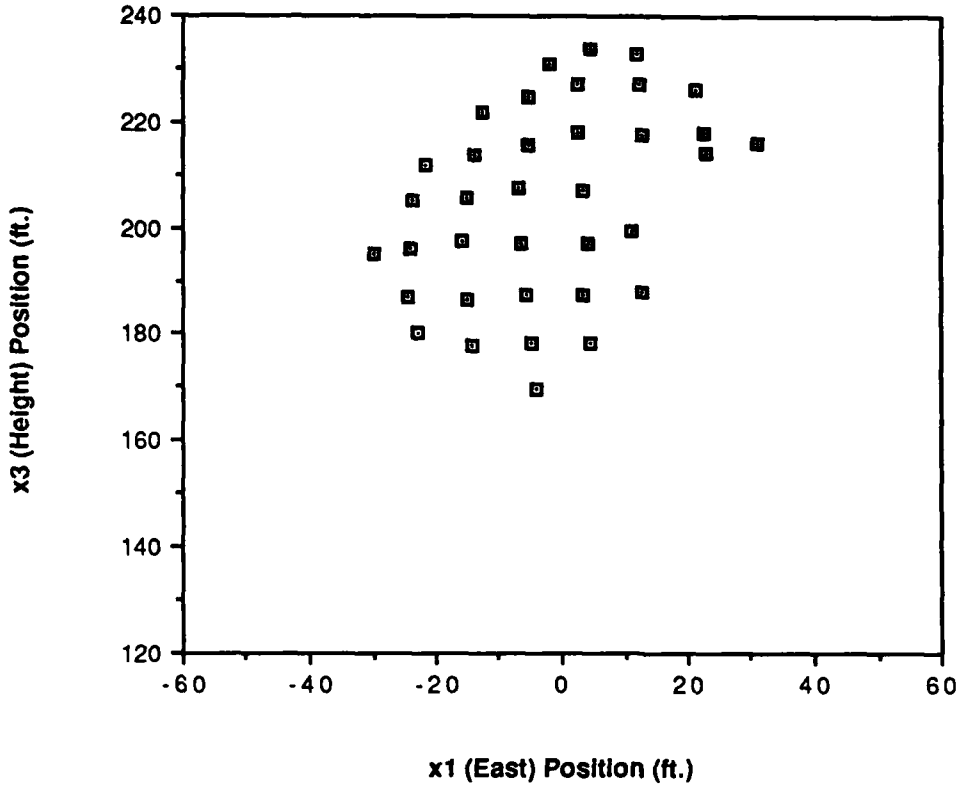


Figure 13.a: Baseline Handprint with CPA under 6 ft.

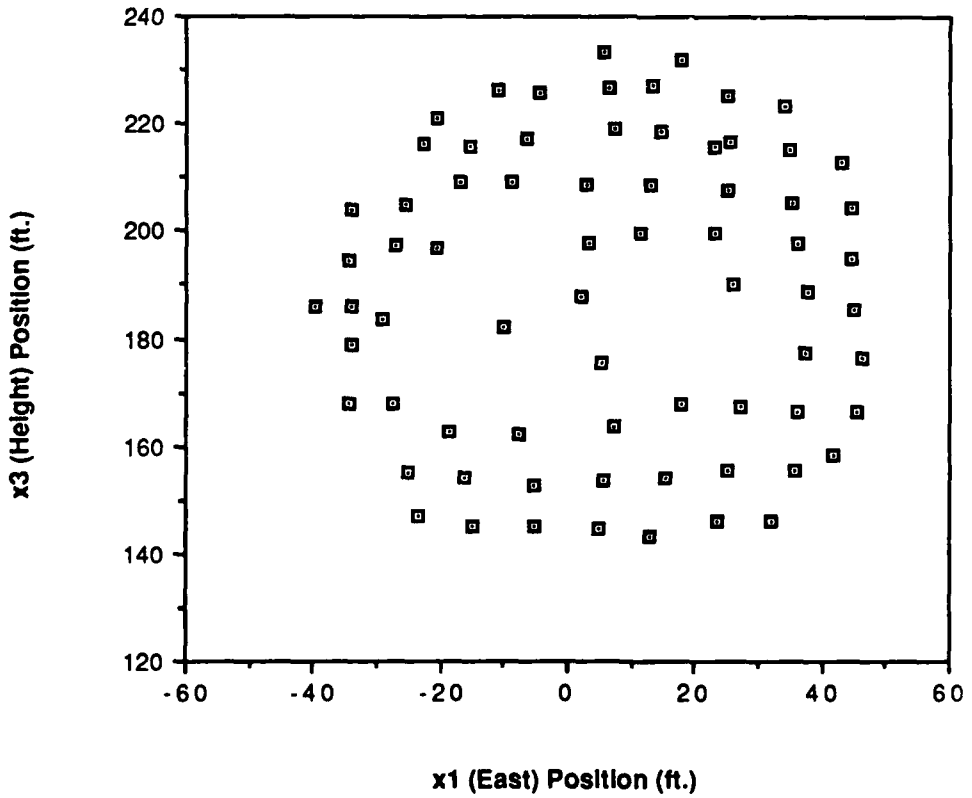


Figure 13.b: Blended Handprint with CPA under 6 ft.

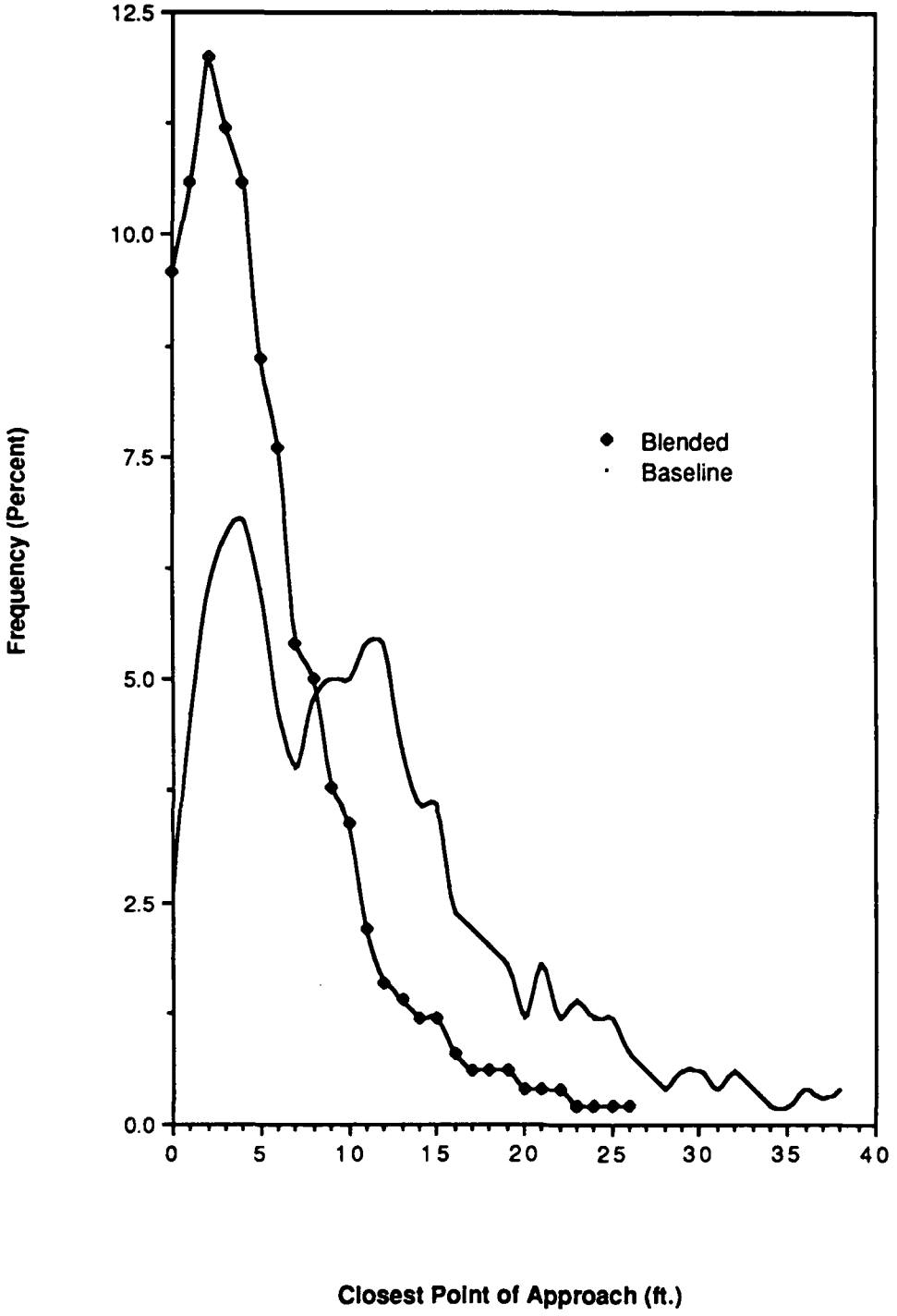


Figure 14: CPA Distribution for Maneuvering Case

Figure 15 presents the accuracy information in another fashion: the x_2 = 10,000 ft. plane is shown riddled by the engagement "bulletholes" (or "projectileholes") which would result from placing each target at the origin. Here, the blended guidance delivers a tight grouping of shots fairly uniformly distributed around the "bullseye", while the baseline guidance is much more spread out and less accurate. The distinctive "X" pattern of the latter results from its diamond-shaped handprint; targets requiring displacement in both the x_1 and x_3 directions are the most difficult for the baseline technique to hit, whereas they cause no particular hardship for an OPTG-based guidance method.

Clearly, a major source of the blended guidance accuracy is its ability to deliver well, over the full region of weapon operability, the divert demanded. Figure 16 compares the ability of the two techniques to track the required divert; the blended law, it can be seen, approaches much better the ideal line of unity slope. In fact, given targets with the maneuver capability as defined above, one can conclude that the OPTG-based method is accurate enough over a wide enough area to hit (i.e., have a CPA of a few feet) any target at which it is correctly (ballistically) aimed -- regardless of the general maneuver direction of the target. Alternately, for non-maneuvering targets, the authority of the OPTG projectile is such to compensate for 4-5 mils of total aiming error in azimuth or elevation.

To summarize the comparison results, it is helpful to define a "ratio of improvement" as the baseline CPA of a given trajectory divided by the CPA resulting from blended guidance. The median such ratio for the maneuvering target case is 1.793. Figure 17 shows the statistic collected into divert bins of five foot width, averaged, smoothed, and plotted onto a log scale against required divert. Note that improvement is as much as an order of magnitude in the intermediate region: 30 to 50 feet. Baseline guidance appears competitive for small diverts from ballistic (up to 20 feet), and both methods begin to run into the physical maneuver limits of the squib-based weapon system beyond about 55 feet. Overall however, the OPTG-based guidance technique is shown to be more accurate than the conventional method at every region of divert.

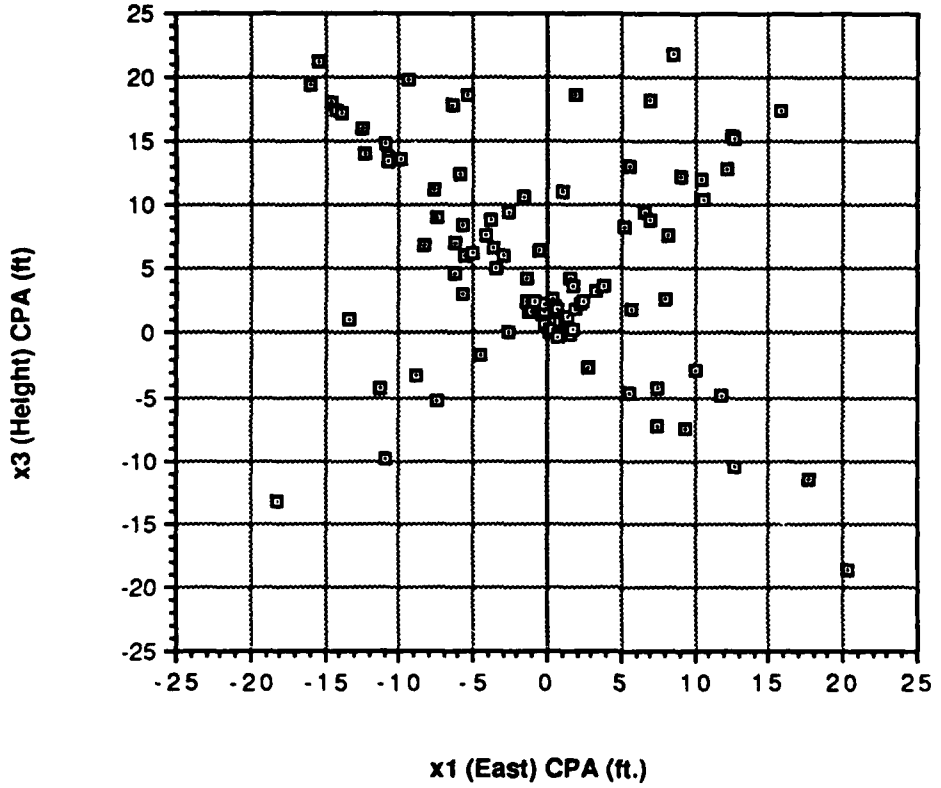


Figure 15.a: Baseline CPA Distribution

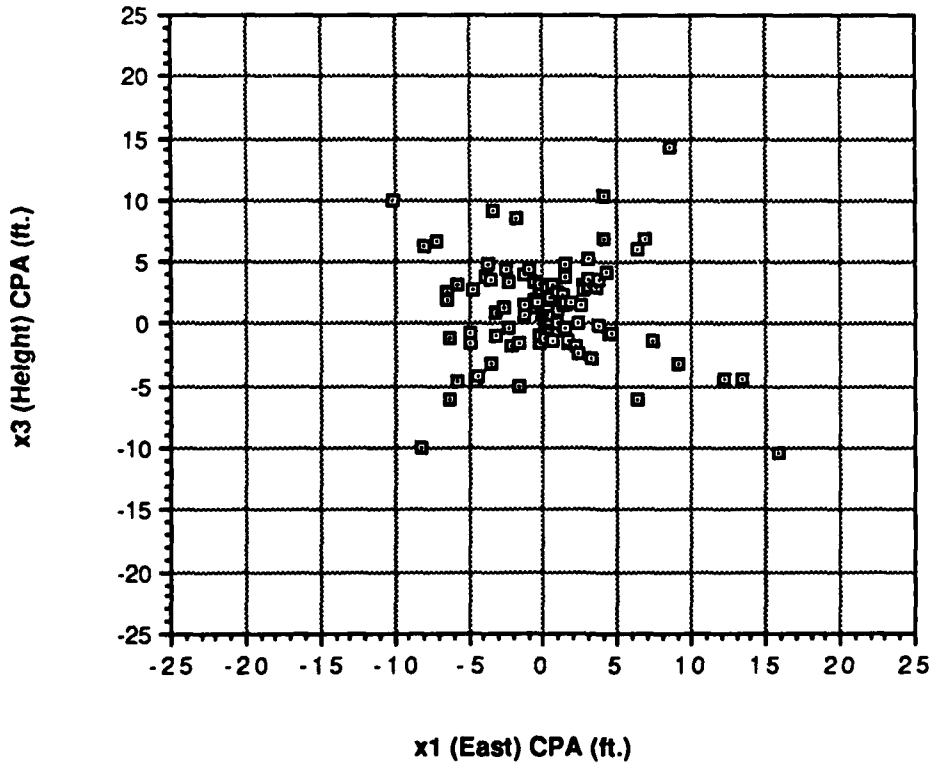


Figure 15.b: Blended CPA Distribution

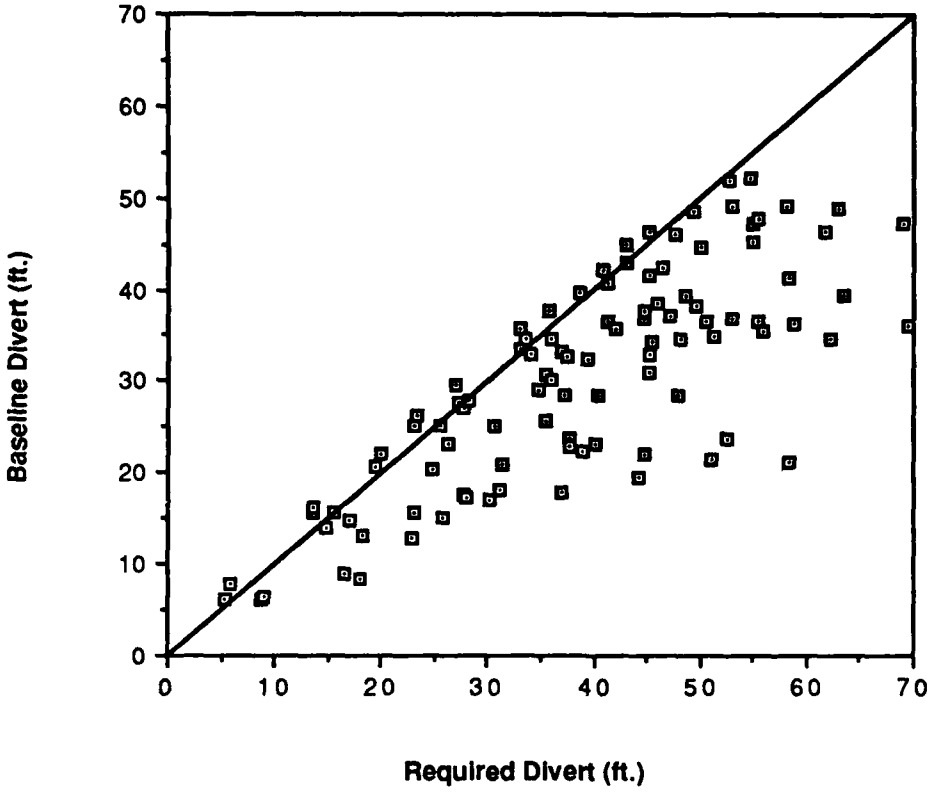


Figure 16.a: Baseline Guidance Divert Tracking

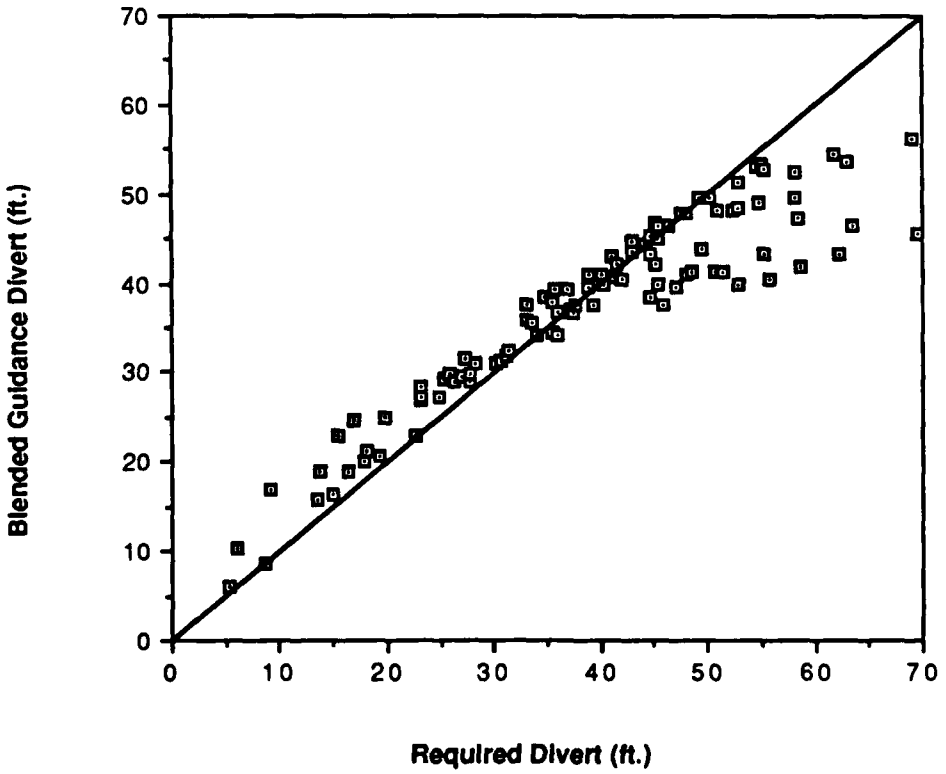


Figure 16.b: Blended Guidance Divert Tracking

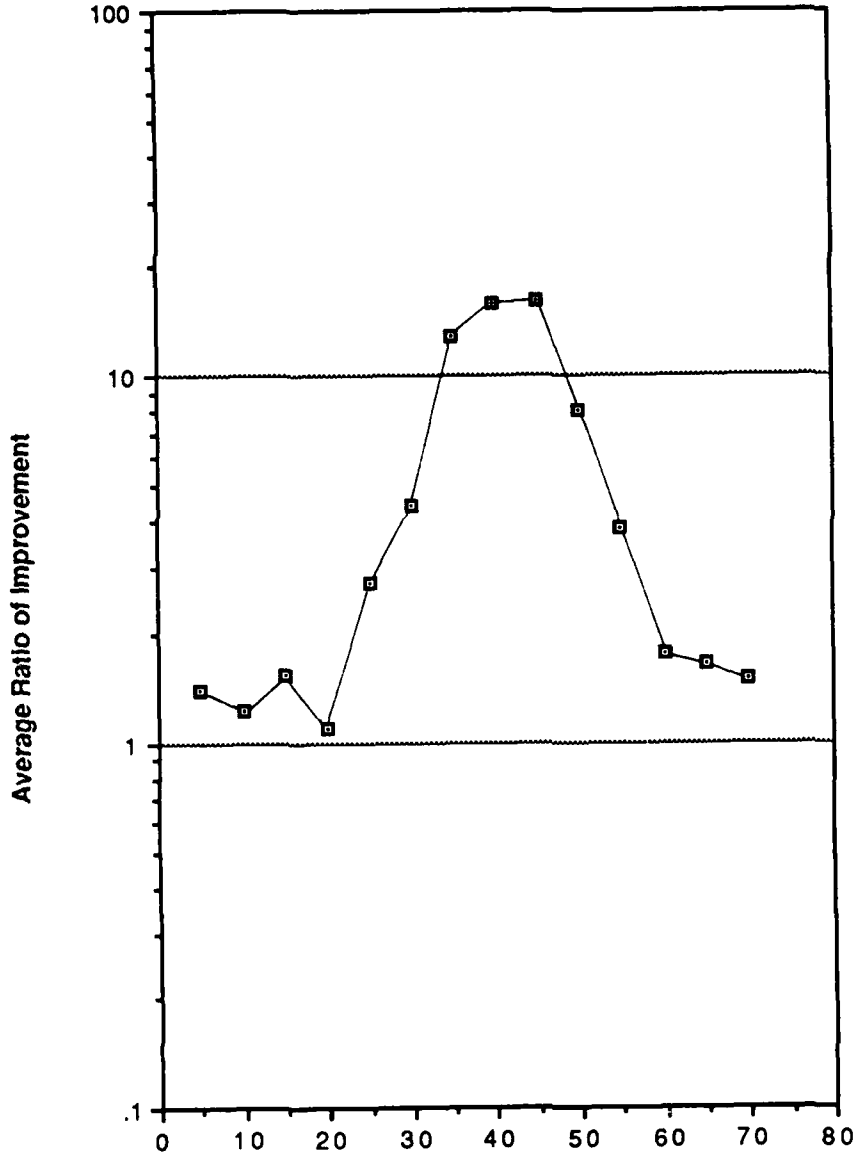


Figure 17: CPA Improvement with Required Divert

7.7 Required Throughput

Each decision timestep, Δt, the OPTG guidance system must update the adjoint differential equations described in Section 4.5. Less often (perhaps one tenth to one one-hundredth of the time) an APN is employed to reinitialize the equations due to new information about the target activity. Summing up the processor requirements of these two components, therefore, provides a good estimate of the throughput required in an OPTG guidance law implementation, exclusive of the tasks of sensor data filtering, target tracking, advanced time-to-go or distance-to-go calculations, or explicit computation of estimated miss distance (if used).

An efficient form of the adjoint equations has been found to consist of 156 multiplies, 91 adds, and 7 look-up operations (the latter for sine, cosine, and square root functions). At a Δt of 0.05 sec., these 254 calculations translate to 5,080 operations per second. The APN of Figure 3.c contains 14 multiplies and 6 adds (in network form), but a more comprehensive APN might consist of up to 50 of each; i.e., 100 operations. Even at a reinitialization interval of 0.50 seconds, such an APN would only contribute 200 operations/sec. to the required throughput, for a total of 5,280 floating-point operations per second. Therefore, a processor capable of only 5.3 kFLOPS (a quite reasonable number) would suffice for the OPTG guidance calculations.

8. REFERENCES AND BIBLIOGRAPHY

1. Lee, R.J., "Generalization of Learning in a Machine", Preprints Papers Presented at 14th Nat'l. Meeting ACM, pp. 21-1 through 21-4, Sept. 1-3, 1959.

2. Gilstrap, L.O., R.J. Lee, "Learning Machines", Bionics Symposium, USAF WADD TR 60-600, pp. 437-450, Sept. 1960.

3. Snyder, R.F., R.L. Barron, and A. Torbett, Advanced Computer Concepts for Intercept Prediction, Vol. I; Conditioning of Parallel Networks for High-Speed Prediction of Re-Entry Trajectories, Adaptronics, Inc. Final Tech. Rept. for U.S. Army Nike-X Proj. Ofc., Redstone Arsenal, AL, Contract DA-36-034-AMC-0099Z, 1964.

4. Kirk, Donald E., Optimal Control Theory, An Introduction, Prentice-Hall, Inc., New Jersey, pg. 151, 1970.

5. Ivakhnenko, A.G., "Polynomial Theory of Complex Systems", IEEE Transactions on SMC, 1, 4, pp. 364-378, Oct. 1971.

6. Barron, R.L., "Adaptive Transformation Networks for Modeling, Prediction, and Control", Proc. IEEE/ORSA Joint Nat'l. Conf. on Major Systems, Anaheim, CA, Oct. 25-26, 1971.

7. Craig, J.N., R.L. Barron, and F.J. Cook, A Priori Training of Guidance and Control Algorithms for Tactical Missiles, Task 1, Air-to-Air Guidance Law Implementation, Adaptronics, Inc., Tech. Rept. AFATL-TR-80-102, Eglin AFB, FL, Sept. 1980.

8. Barron, R.L., A.N. Mucciardi, F.J. Cook, J.N. Craig, and A.R. Barron, "Adaptive Learning Networks: Development and Application in the United States of Algorithms Related to GMDH", Self-Organizing Methods in Modeling: GMDH Type Algorithms, S.J. Farlow (Ed.), Marcel Dekker, Inc., New York, Chap. 2, pp. 25-65, 1984.

9. Hutchings, Thomas D., "Prediction and Control Algorithms for Correctable Projectiles", Research in Fire Control and Small Caliber Armaments, U.S. Army Armament Research and Development Center, SCS-C-IR(SR) 83-005, January 1984, pp. 2-35 to 2-43.

10. Hutchings, Thomas D., "Prediction and Control Algorithms for Correctable Projectiles", Research in Fire Control and Small Caliber Armaments, U.S. Army Armament Research and Development Center, SCS-C-IR(SR) 84-007, December 1984, pp. 2-21 to 2-30.

11. Hutchings, Thomas D., "Prediction and Control Algorithms for Correctable Projectiles", Research in Fire Control and Small Caliber Armaments, U.S. Army Armament Research and Development Center, SCS-C-IR(SR) 85-002, December 1985, pp. 2-23 to 2-27.

12. Elder, J.F. IV, S.R. Goldschmidt, D.W. Barron, D. Misra, User's Manual: ASPN -- Algorithm for Synthesis of Polynomial Networks, Barron Associates, Inc. Item 2 Final Rept. for Century Computing, Inc. under P.O. 5035, F33615-84-C-3609, Nov. 1986.

13. Barron, Roger L. and Paul Hess, Development of Air Interceptor Two-Point Boundary-Value Guidance Based upon Adaptively-Synthesized Polynomial Networks, Final Report, Barron Associates, Inc., December 31, 1986.

14. "Scope of Work for Fire Control Guided Air Defense/Anti-Armor Heat Projectile", U.S. Army Armament Research and Development Center, 1987.

15. Abbott, Dean W., Extensions in the Optimization of Two-Point Boundary Value Problems Utilizing the Calculus of Variations, Master's Project Report, University of Virginia, May 7, 1987 (revision in progress).

16. Shrier, S., R.L. Barron, and L.O. Gilstrap, "Polynomial and Neural Networks: Analysis and Engineering Applications", Proc. IEEE First Int'l. Conf. on Neural Networks, San Diego, CA, June 21-24, 1987, Vol. II, pp. 431-439.

17. Barron, R.L., "Optimum, Real-Time Two-Point Boundary-Value Guidance for Tactical Systems", Invited Seminar, U.S. Naval Weapons Center, presented in behalf of HR Textron Inc., June 23, 1987.

18. Elder, John F., IV and Roger L. Barron, Uses of Inductive Modeling Tools for Design of Reconfigurable Flight Control Systems, Barron Associates, Inc. Final Report under Purchase Order 5238 for Century Computing Inc., U.S. Air Force Wright Aeronautical Laboratories Contract F33615-84-C-3609, July 1987.

19. "Use of Polynomial Networks to Improve on Control Efficiency of Maneuverable Projectile", Contract DAAA21-87-C-0140 between Department of Army, AMCCOM, Picatinny Arsenal, NJ, and Barron Associates, Inc., July 29, 1987.

20. Barron, Roger L., Performance Criterion Specification for Optimum Path-to-Go Guidance of Tactical Air-to-Surface Glide Weapons, Barron Associates, Inc. Interim Report No. 1 under Purchase Order B4789 for HR Textron Inc., U.S. Naval Weapons Center Contract N60530-88-C-0036, July 31, 1987.

21. Barron, Roger L., Dean W. Abbott, and John F. Elder IV, Derivation and Use of Variational Equations for Optimum Path-to-Go Guidance of Tactical Air-to-Surface Glide Weapons, Barron Associates, Inc. Interim Report No. 3 under Purchase Order B4789 for HR Textron Inc., U.S. Naval Weapons Center Contract N60530-88-C-0036, August 15, 1987.

22. Goldschmidt, Stephen R., Roger L. Barron, John F. Elder IV, and Dean W. Abbott, Optimum Path-to-Go Guidance of a Standard-Geometry Mk 82 Glide Weapon, Barron Associates, Inc. Interim Report No. 4 under Purchase Order B4789 for HR Textron Inc., U.S. Naval Weapons Center Contract N60530-88-C-0036, September 8, 1987.

23. Abbott, Dean W., Roger L. Barron, Optimum Performance of the Skipper Boost-Glide Weapon, Barron Associates, Inc. Interim Report No. 5 under Purchase Order B4789 for HR Textron Inc., U.S. Naval Weapons Center Contract N60530-88-C-0036, January 22, 1988.

APPENDIX A

A. APPROXIMATE EQUATIONS OF TRANSLATIONAL MOTION FOR CAT PROJECTILE

The three-degree-of-freedom translational equations of motion for the CAT projectile, treated as a mass particle, are, in a wind-axes coordinate system, using a flat-Earth approximation and ignoring the effects of winds:

$$m\dot{V} + mg \sin\gamma + q S C_D + T(\cos\varphi \sin\alpha_W - \sin\varphi \sin\beta_W) = 0 \quad \text{A:1}$$

$$mV\dot{\psi} \cos\gamma - q S C_{Y_W} - T \sin\varphi \cos\beta_W = 0 \quad \text{A:2}$$

$$mV\dot{\gamma} + mg \cos\gamma - q S C_{L_W} - T \cos\varphi \cos\alpha_W = 0 \quad \text{A:3}$$

in which:

m = mass (here assumed constant)

V = velocity magnitude, measured tangential to instantaneous flight path

g = acceleration of gravity (here assumed constant)

γ = flight path elevation angle, measured positive up from horizontal plane

q = dynamic pressure $= \frac{1}{2}\rho V^2$

ρ = atmosphere density (here assumed constant)

S = reference area

C_D = coefficient for component of aerodynamic drag force acting along negative of velocity vector

C_{L_W} = coefficient for component of aerodynamic normal force acting perpendicular to the velocity vector in the vertical plane containing $\vec{V}$, measured positive up

C_{Y_W} = coefficient for component of aerodynamic normal force acting in a horizontal direction that increases ψ

T = thrust perpendicular to projectile body $\vec{x}$ axis (spin axis), acting at roll attitude angle φ

φ = projectile roll attitude angle, measured clockwise from the vertical plane containing $\vec{x}$ (as viewed from rear of projectile)

α_W = aerodynamic angle of attack in wind-axes system

β_W = aerodynamic angle of sideslip in wind-axes system

ψ = flight path heading angle, measured $\vec{x}_3$ (verticle) axis

$()_W$ = denotes the wind-axes system

For small α_W and β_W, the lift and side-force coefficients may be expressed as:

$$C_{L_W} = C_{N\alpha}\,\alpha_W - C_1 \frac{P}{V}\beta_W \qquad \text{A:4}$$

$$C_{Y_W} = -\,C_{N\alpha}\,\beta_W - C_1 \frac{P}{V}\,\alpha_W \qquad \text{A:5}$$

where:

$C_{N\alpha}$ = aerodynamic normal force coefficient (here assumed constant)

C_1 = Magnus force coefficient $\equiv C_{Y_{P\alpha}}\, d_{ref}\, /2$ (here assumed constant)

d_{ref} = reference length

P = rolling rate, measured positive in RH direction along projectile body $\vec{x}$ axis, i.e., positive in direction of increasing φ (P is here assumed constant)

The drag coefficient may be written as a quadratic drag polar:

$$C_D = C_{D_0} + K\left(C_{L_w}^2 + C_{Y_w}^2\right) \qquad \text{A:6}$$

where:

C_{D_0} = drag coefficient when lift and side forces are zero (here C_{D_0} is assumed constant)

K = induced-drag factor (here assumed constant)

Substituting Eqs. A:4 - A:6 into A:1 - A:3, one obtains, for the case in which α_w and β_w are small:

$$m\dot{V} + mg\sin\gamma + qS\left[C_{D_0} + K\left(C_{N_\alpha}^2 + C_1^2\frac{P^2}{V^2}\right)\left(\alpha_w^2 + \beta_w^2\right)\right] + T\left(\alpha_w\cos\varphi - \beta_w\sin\varphi\right) = 0 \qquad \text{A:7}$$

$$mV\dot{\psi}\cos\gamma + qS\left(C_{N_\alpha}\beta_w + C_1\frac{P}{V}\alpha_w\right) - T\sin\varphi = 0 \qquad \text{A:8}$$

$$mV\dot{\gamma} + mg\cos\gamma - qS\left(C_{N_\alpha}\alpha_w - C_1\frac{P}{V}\beta_w\right) - T\cos\varphi = 0 \qquad \text{A:9}$$

Also, for small α_w and β_w:

$$\alpha_w = \alpha_{tot}\cos\varphi \qquad \text{A:10}$$

$$\beta_w = -\alpha_{tot}\sin\varphi \qquad \text{A:11}$$

where α_{tot} is the total aerodynamic angle.

In Ref. 2 it is reported that the firing of a squib in the CAT projectile produces two temporary effects: a brief reaction force, T, of duration δt (a few milliseconds) and a transient aerodynamic force. When the transient forces have decayed, "...the resulting steady-state solution to the change in heading from lift force effects is given by

$$\Delta\theta = \frac{qS}{2mV} C_{N\alpha}\, \alpha_{max} \left(\frac{1}{\omega_1} - \frac{1}{\omega_2} \right)" \qquad \text{A:12}$$

in which α_{max} is the maximum value of the damped oscillation, ω_1 is the nutation frequency (the frequency of the projectile short-period oscillation), and ω_2 is the precession frequency.

But $mV\Delta\theta$ is the change in momentum of the particle and is equal to the effective lift force multiplied by a time interval Δt established by the time required for the transient aerodynamic force to subside. However, the effective lift force may be expressed as $q\, S\, C_{N\alpha}\, \alpha_{eff}$, where α_{eff} is the effective angle of attack over the interval Δt due to the thruster impulse. Therefore

$$q\, S\, C_{N\alpha}\, \alpha_{eff}\, \Delta t = mV\Delta\theta \qquad \text{A:13}$$

and

$$\alpha_{eff} = \frac{mV\Delta\theta}{q\, S\, C_{N\alpha}\, \Delta t} \qquad \text{A:14}$$

From Ref. 2

$$\alpha_{max} = \frac{I_s\, l_T \exp\left(\dfrac{-\pi\zeta_1/2}{\sqrt{1-\zeta_1^2}} \right)}{I_{yy}\, \omega_1 \sqrt{1-\zeta_1^2}} \qquad \text{A:15}$$

wherein:

I_s = squib total impulse (approximately 0.75 lb.-sec.) = $T\, \delta t$

l_T = thrust moment arm, measured positive ahead of C.G.

I_{yy} = pitching moment of inertia of projectle

ζ_1 = nutational damping coefficient

Combining Eqs. A:12, A:14, and A:15:

$$\alpha_{eff} = \frac{T\,\delta t\, l_T \left(\frac{1}{\omega_1} - \frac{1}{\omega_2}\right) \exp\left(\frac{-\pi\zeta_1/2}{\sqrt{1-\zeta_1^2}}\right)}{2\Delta t\, I_{yy}\, \omega_1 \sqrt{1-\zeta_1^2}} \qquad \text{A:16}$$

Let

$$T^* = \text{scaled thrust} \equiv T\,\delta t/\Delta t << T \qquad \text{A:17}$$

and

$$\alpha_{eff} = C_0 T^* \qquad \text{A:18}$$

where

$$C_0 \equiv \frac{l_T \left(\frac{1}{\omega_1} - \frac{1}{\omega_2}\right) \exp\left(\frac{-\pi\zeta_1/2}{\sqrt{1-\zeta_1^2}}\right)}{2\, I_{yy}\, \omega_1 \sqrt{1-\zeta_1^2}} \qquad \text{A:19}$$

For present purposes, C_0 is assumed to be constant. But α_{eff} may be viewed as the total aerodynamic angle during the interval (t, t + Δt), and thus Eqs. A:10, A:11 become, for the case in which a squib is fired at the beginning of the interval:

$$\alpha_w = C_0 T^* \cos\varphi \qquad \text{A:20}$$

$$\beta_w = -\, C_0 T^* \sin\varphi \qquad \text{A:21}$$

Having accounted for the aerodynamic effects of a squib firing, and noting that the reaction effects are those of a scaled thrust T^* acting over the entire interval $(t, t + \Delta t)$, one has (from Eqs. A:7 - A:9) the following two sets of equations of motion.

When a squib is not fired

$$m\dot{V} + mg \sin\gamma + q S C_{D_0} = 0 \qquad \text{A:22}$$

$$mV\dot{\psi}\cos\gamma = 0 \qquad \text{A:23}$$

$$mV\dot{\gamma} + mg\cos\gamma = 0 \qquad \text{A:24}$$

When a squib is fired

$$m\dot{V} + mg\sin\gamma + q S C_{D_0} + C_0(T^*)^2 \left[q SK\left(C_{N_\alpha}^2 + C_1^2\frac{P^2}{V^2}\right)C_0 + H_1\right] = 0 \qquad \text{A:25}$$

$$mV\dot{\psi}\cos\gamma + T^*\left(q S C_0 H_3 - \sin\varphi\right) = 0 \qquad \text{A:26}$$

$$mV\dot{\gamma} + mg\cos\gamma + T^*\left(q S C_0 H_5 - \cos\varphi\right) = 0 \qquad \text{A:27}$$

wherein:

$$H_1 \equiv 1 \qquad \text{A:28}$$

$$H_3 \equiv -C_{N_\alpha}\sin\varphi + C_1\frac{P}{V}\cos\varphi \qquad \text{A:29}$$

$$H_5 \equiv -C_{N_\alpha}\cos\varphi - C_1\frac{P}{V}\sin\varphi \qquad \text{A:30}$$

For present purposes, we shall employ the following parameters, treating them as constants. In future work, these may be taken as functions of the indicated variables:

$$C_2 \equiv C_1 \frac{P}{V} = C_{Y_{P\alpha}} d_{ref} \left(\frac{P}{2V}\right) \qquad \text{A:31}$$

$$C_3 \equiv K\left(C_{N\alpha}{}^2 + C_2{}^2\right) C_0{}^2 \qquad \text{A:32}$$

Next, let $\dot{N}$ be a variable that signifies if a thruster is fired within (i.e., at the beginning of) a solution interval $(t, t + \Delta t)$. Then $\dot{N}$ is binary, $\dot{N} = 0$ indicating no firing, $\dot{N} = 1$ indicating that a firing occurs. Now, the equations of motion (A:22 - A:27) become:

$$m\dot{V} + mg \sin\gamma + q S C_{D_0} + (T^*)^2 \left(q S C_3 + C_0 H_1\right) \dot{N} = 0 \qquad \text{A:33}$$

$$mV \dot{\psi} \cos\gamma + T^* \left(q S C_0 H_3 - \sin\varphi\right) \dot{N} = 0 \qquad \text{A:34}$$

$$mV \dot{\gamma} + mg \cos\gamma + T^* \left(q S C_0 H_5 - \cos\varphi\right) \dot{N} = 0 \qquad \text{A:35}$$

In Eq. A:33, $\dot{N}$ is not squared because its influence on the thrust terms is the same whether used as $\dot{N}$ or $\dot{N}^2$.

APPENDIX B

B. COEFFICIENTS IN PROJECTILE DRAG POLAR

The drag polar of a projectile is the series expansion of its wind-axes drag coefficient (C_D) as a function of its wind-axes lift coefficient (C_L):

$$C_D = C_{D0} + |C_L| \, \partial C_D/\partial C_L + C_L{}^2 \partial C_D/\partial C_L{}^2 + \ldots \qquad \text{B:1}$$

Typically:

$$C_D = C_{D0} + C_{D\alpha 2} \, \alpha^2 + \ldots \qquad \text{B:2}$$

$$C_L = C_{L\alpha} \alpha + C_{L\alpha 3} \, \alpha^3 + \ldots \qquad \text{B:3}$$

where α is here used to represent the effective total aerodynamic angle of attack.

Wind-tunnel data are customarily expressed in the form of projectile body-axes force coefficients. As shown by Hutchings* partial derivatives in the wind-axes system may be written in terms of partial derivatives in body axes, viz.:

$$C_{D0} = C_{X0} \qquad \text{B:4}$$

$$C_{D\alpha^2} = C_{N\alpha} - \frac{1}{2} C_{X0} + C_{X\alpha^2} \qquad \text{B:5}$$

$$C_{L\alpha} = C_{N\alpha} - C_{X0} \qquad \text{B:6}$$

$$C_{L\alpha 3} = C_{N\alpha 3} - C_{X\alpha^2} - \frac{1}{2} C_{N\alpha} \qquad \text{B:7}$$

where the subscripts X and N denote axial (rearward) and normal force components, respectively.

---

* Hutchings, Thomas D., notes dated September 13, 1983.

Now:

$$\partial C_D/\partial C_L = \frac{\partial C_D/\partial \alpha}{\partial C_L/\partial \alpha} = \frac{C_{D\alpha}}{C_{L\alpha}} = 0 \qquad \text{B:8}$$

$$\partial C_D/\partial C_L^2 = \frac{\partial C_D/\partial \alpha^2}{\partial C_L^2/\partial \alpha^2} = \frac{C_{D\alpha^2}}{\partial C_L^2/\partial \alpha^2} \qquad \text{B:9}$$

But, from Eq. B:3:

$$C_L^2 = (C_{L\alpha})^2 \alpha^2 + \ldots \qquad \text{B:10}$$

whence

$$\partial C_L^2/\partial \alpha^2 = (C_{L\alpha})^2 \qquad \text{B:11}$$

and, therefore

$$\partial C_D/\partial C_L^2 = \frac{C_{D\alpha^2}}{(C_{L\alpha})^2} \qquad \text{B:12}$$

Defining

$$K \equiv \partial C_D/\partial C_L^2 = \frac{C_{D\alpha^2}}{(C_{L\alpha})^2} = \text{induced-drag factor} \qquad \text{B:13}$$

and introducing the quadratic-form drag polar

$$C_D = C_{D0} + KC_L^2 \qquad \text{B:14}$$

one then has C_{D0} determined by Eq. B:4 and

$$K = \frac{C_{N\alpha} - \frac{1}{2}C_{X0} + C_{X\alpha^2}}{(C_{N\alpha} - C_{X0})^2} \qquad \text{B:15}$$

APPENDIX C

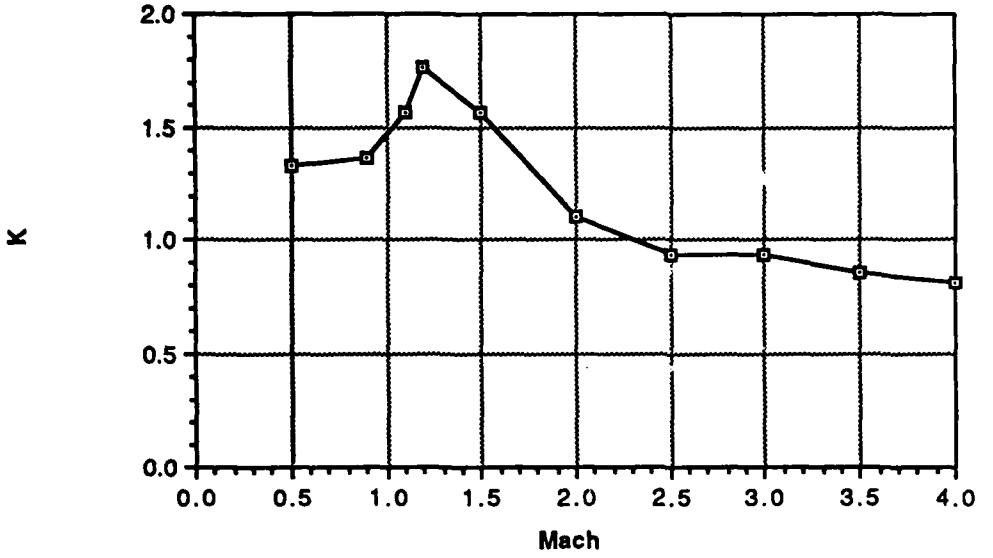
C. PROJECTILE AERODYMAMIC COEFFICIENT VALUES

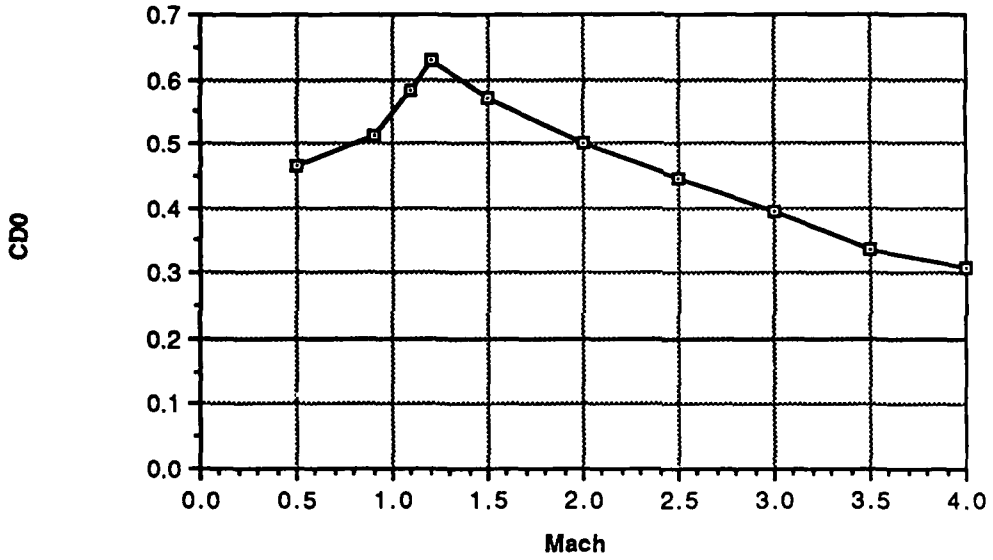
The following aerodynamic coefficient values have been calculated from ARDEC data provided December 10, 1987:

M	C_{D_0}	K	$C_{N\alpha}$, rad.$^{-1}$
0.5	0.467	1.335	5.03
0.9	0.514	1.362	5.03
1.1	0.582	1.569	5.03
1.2	0.631	1.768	5.03
1.5	0.572	1.562	5.03
2.0	0.500	1.110	5.55
2.5	0.444	0.940	5.61
3.0	0.394	0.942	5.55
3.5	0.336	0.858	5.35
4.0	0.308	0.812	5.01

The above coefficient values are based upon a reference area (S) of 0.12151 ft.^{2}. The induced-drag factor (K) decreases with α, indicating that the drag polar for this projectile is not a pure parabola. Because α_{eff} (Eq. A:18) is a small fraction of 1 deg., we have used ARDEC data for $\alpha = 0$ and $\alpha = 1$ deg., rather than ARDEC data for $\alpha = 0$ and $\alpha = 2$ deg., in computing K.

Figure C.1 shows the dependencies of K, C_{D_0}, and $C_{N\alpha}$ on Mach number.





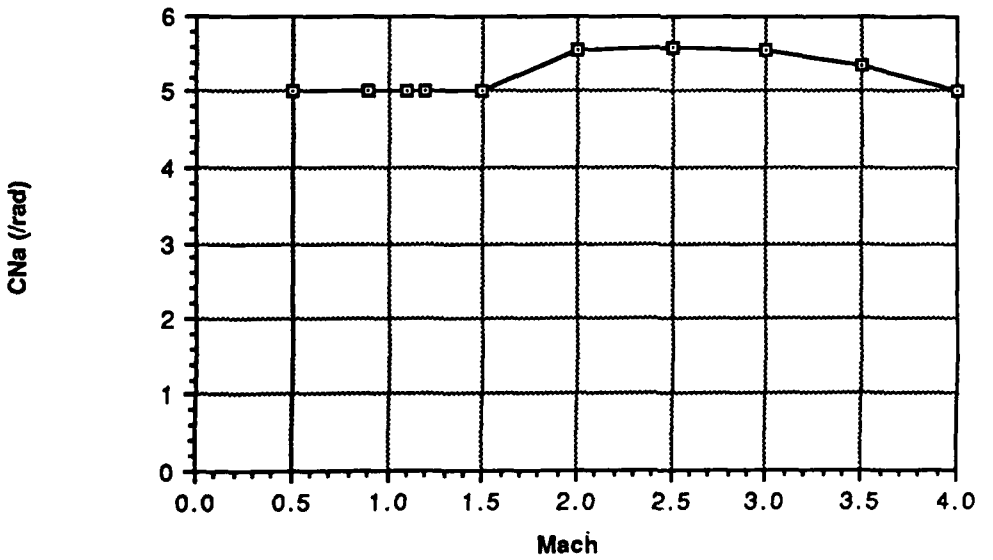


Figure C.1: Dependencies of K, C_{D_0}, and C_{N_α} on Mach Number

APPENDIX D

D. ALTERNATIVE ESTIMATE OF TIME TO GO*

It has been noted that

$$\beta' \equiv \frac{\rho S C_{D_0}(M) V^{1/2}}{4m} \qquad \text{D:1}$$

is approximately constant over the Mach number range of interest ($1.2 \leq M \leq 3.5$), and it can therefore be shown that

$$\tau \approx \frac{R_{go}}{V - \beta' V^{1/2} R_{go}} \qquad \text{D:2}$$

where R_{go} is the distance to go to reach the target.

For the projectile described in Appendix C, with m = 0.966 slug, β' calculated using sea level standard atmosphere density is 1.695×10^{-3}. This gives close agreement with the actual τ values during OPTG guidance.

* Suggested by Albert Rahe of ARDEC, December 10, 1987.

APPENDIX E

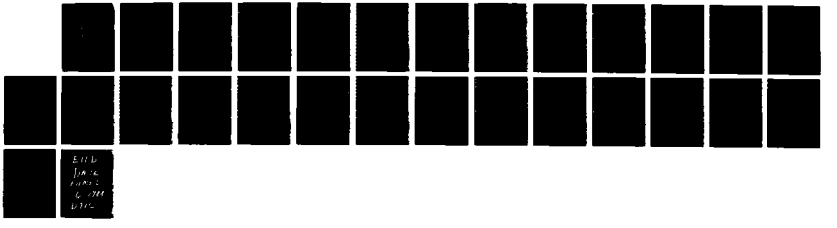
E. SOURCE LISTING OF SIMULATION PROGRAM

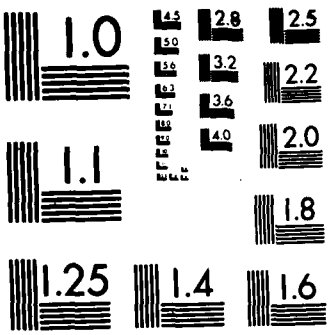
The critical components of the 3DOF simulation are listed here. As the program is in the "C" language, the file in which a routine appears is significant, so a brief directory of routines and their locations is provided. The "makefile", appearing last (page 115), reveals the file dependencies and automatically governs program re-compilation.

```
/*  main.c      */

#include "standard.h"                   /*  standard functions, consts  */
 /*  (stdio.h, math fns, deg, rad, nl)  */
#include "integrate.h"                  /*  integration structure       */
 /*  (STATE_VAR, N_STATES, St(), LOOP_St(), FOR3())                      */
#include <interface1.h>                 /*  interface strct DESCRIPTION */
#include <GetfilePkg.h>                 /*  GETFILE, FP, GFILE, gclose  */
#include <screen.h>                     /*  wide, unwide fns            */


 /*  Global variables used in dynamics  */
 /*  state variables                    */
 double phi;                            /*  rotation angle (rad)        */
 double x[4];                           /*  position vector (ft) (1-3)  */
 double V, gamma, psi;                  /*  velocity vector (ft/s, rad) */
 double lam[13];                        /*  lambdas (1-12; 4-6 const)   */
 double xT[4], vT[4];                   /*  target position, velocity   */

 /*  pseudo-state variables             */
 double P;                              /*  roll rate (deg/sec)         */
 double W[9];                           /*  guidance parameters (0-8)   */
 double SW[4];                          /*  guidance params, fn(W6,7,8) */
 int N;                                 /*  number of squibs fired      */
 double N_dot;                          /*  control variable            */
 double uT[4];                          /*  target acceleration (E,N,H) */
 double Umean, Usig;                    /*  T acceleration noise params */

 double Range, Tau;                     /*  range, time to target (dyn) */
 int hunting, S_last;                   /*  dt iteration vars (fly)     */
 int squib;                             /*  flags squib firing (fly)    */
 int Abort;                             /*  halt flag                   */


 /*  (Primarily local) interface vars   */
 double duration;                       /*  max flight duration (sec)   */
 double dt0;                            /*  original time step (sec)    */
 int debug;         double ddebug;      /*  debug flag (used everywhere)*/
 int bounce;        double dbounce;     /*  "bounce" flag               */
 int G_mode;        double dG_mode;     /*  guidance mode (0,1,2)       */
 int R_mode;        double dR_mode;     /*  recording mode (0,1,2)      */
 int S_mode=FALSE;  double dS_mode;     /*  search mode (0,1,2,3,4,5)   */
 /* (S_mode used before interface call) */
 int wide;          double dwide;       /*  wide screen print option    */
 int p_step;        double print_int;   /*  print interval (iter, sec)  */
 int hist;          double dhist;       /*  store var histories flag    */
                    double dN_f;        /*  final N (backward only)     */
 int integ;         double dinteg;      /*  integration (0:rec; 1:Heun) */
 double Nd_thresh;                      /*  firing threshold            */
#include "init.h"                       /*  interface information       */
 /* (fwdinfo, backinfo; included after declarations & interface include)*/
```

AD-A193 377 OPTIMUM-PATH-TO-GO GUIDANCE OF COMMAND ADJUSTED TRAJECTORY PROJECTILES(U) BARRON ASSOCIATES INC STANARDSVILLE VA R L BARRON ET AL. 22 JAN 88 129-F 2/2

UNCLASSIFIED DAAA21-87-C-0140 F/G 17/7 NL





```
 /*  (primarily) local shared variables */
 FILE *fp, *vfp;                        /*  record, publish file ptrs   */
 GFILE *hp;                             /*  gfile ptr: variable history */
 double Time, dt;                       /*  current time (cap!), step   */
 int iter, trials;                      /*  # iterations, trials        */
 int fwd;                               /*  flight direction flag       */
 double random(), r_time;               /*  random fn, record time      */
 double score;                          /*  trial performance (grope)   */
 char str[SCR_WIDTH];                   /*  reply string                */


main()                                  /*  129-00 simulation program   */
{
 STATE_VAR state[N_STATES];             /*  state variables (integrated)*/
 int getdir();                          /*  flight direction fn         */
 int more_trials(), conds_allow();      /*  control routines            */
#define maxVARS  44                     /*  max # interface variables   */
 double bstore[maxVARS];                /*  bkwd interface var storage  */
 double fstore[maxVARS];                /*  fwd  interface var storage  */
 int btrials=0, ftrials=0;              /*  # trials flown in each dir  */
 FILE *ffp, *bfp;                       /*  fwd, bkwd file ptrs (see fp)*/
 double *storage, *sptr;                /*  storage pointers            */
 struct DESCRIPTION *info, *dptr;       /*  ptrs to interface structure */
 int wscreen, tmp;                      /*  screen flag, work integer   */

#define BANNER(h)  printf("\n=====  %s  =====\n", h)
#define SPR(h)  sprintf(str, ((h) ? "Launch %d" : "Impact %d"), trials)
#define ENDnote(h)  if (!Abort) { SPR(h); BANNER(str); record(state); }
/* ---------------------------------------------------------------------- */

   /*  un-wide the screen (to put all runs on common footing)   */
   scrUnwide; wscreen = FALSE;  /* (& track screen condition)   */

   while( getdir() )  { /*  get direction flag (fwd) for series */
      /*  set direction-dependent interface variables   */
      if (fwd)  {  storage=fstore;  info= fwdinfo;  trials=ftrials;  }
      else      {  storage=bstore;  info=backinfo;  trials=btrials;  }
      printf( "(#trials flown in that direction = %d)\n", trials );

      /*  open data base file only if going in that direction   */
      if (!trials)  if (fwd)  ffp = fopen( "impact.db", "w" );
                    else      bfp = fopen( "launch.db", "w" );
      fp = (fwd ? ffp : bfp);           /*  set file pointer    */

      while( more_trials( fwd, trials, info ) )  {
         /*  store interface variables for recall       */
         for ( sptr=storage, dptr=info;  dptr->var!=NULL;
             *(sptr++) = *( (dptr++)->var ) )  ;

         /*  translate interface variables into internal units  */
         tmp = (int)(print_int/dt0);  p_step = max(1, tmp);
         G_mode = (int)dG_mode;  wide = (int)dwide;  debug  = (int)ddebug;
         R_mode = (int)dR_mode;  hist = (int)dhist;  bounce = (int)dbounce;
         S_mode = (int)dS_mode;  integ= (int)dinteg;
```

```
/*  translate or initialize/finalize   */
if (fwd) { psi *= DEGtoRAD;  phi *= DEGtoRAD;  N = 0;          }
else     { psi  = 0.0;       phi  = 0.0;       N = (int)dN_f;  }

/*  Change screen width if wrong; open history file if asked   */
if      ( wide && !wscreen)  { scrWide;    wscreen=TRUE;  }
else if (!wide &&  wscreen)  { scrUnwide;  wscreen=FALSE; }
if (hist)  {
   if (!(vfp = GETFILE( hp, "flight data file:", "var.hist", "w")) )
      { printf("?Couldn't open var history file.\n");  hist=FALSE; }
}

/*  Reset time variables, position, counters, flags, constants */
if (fwd)  { dt =  dt0;  Time = 0.0;       FOR3( x[k] =   0.0; ) }
else      { dt = -dt0;  Time = duration;  FOR3( x[k] = xT[k]; ) }
tconst();                          /*  squib constants (fn(P,dt0)) */
iter    = 0;    Abort = FALSE;          /*  start of iterations */
S_last  = 1;  hunting = FALSE;          /*  don't hunt 'til end */
N_dot = 0.0;    squib = FALSE;          /*  default:  no firing */
if (R_mode==1)  r_time = random( 2*dt,1.0 );   /*  recording 1 */

if (fwd) {                         /*  set direction weights       */
   SW[1] = -W[6]*sin(psi) - W[7]*cos(psi);     /*  for lamdot10*/
   SW[2] = -W[6]*cos(psi) + W[7]*sin(psi);     /*  for lamdot11*/
   SW[3] = W[8];                               /*  for lamdot12*/
}
else  FOR3( SW[k] = 0.0; )     /*xx(not ready to use SW bkwd)  */

/*  Establish integration structure states and derivatives      */
if (!fwd)  finalize();                  /*  finalize V, lambdas */
/*  sequester states; calculate (& seq.) resulting derivatives */
set_st(state);  deriv(state);  /* (also N_dot,squib,Range,Tau) */

ENDnote( fwd )
while ( conds_allow() )  fly( state );
ENDnote(!fwd )  publish();

/*  evaluate trial (for grope) */
if (fwd)  score = Range;        /*  minimize final range        */
else      score = x[2];         /*  maximize downrange (N<Nf)   */

if (bounce>=0)  {                       /*  bouncing flight!    */
   fwd = !fwd;  tmp=G_mode;             /*  reverse & save flag */
   G_mode = bounce;                     /*  use bounce guidance */
   sprintf(str, "Bounce guidance mode: %d", G_mode );  BANNER( str );

   iter    = 0;    Abort = FALSE;       /*  start of iterations */
   S_last  = 1;  hunting = FALSE;       /*  don't hunt 'til end */
   N_dot = 0.0;    squib = FALSE;       /*  default:  no firing */
   /* xx(these 3 init lines are repeats; could become a func)  */

   /*  stutter-step to get back on dt0 multiples       */
   dt = (int)(Time/dt0)*dt0 - Time;  if (fwd) dt += dt0;  fly( state );
   dt = (fwd ? dt0 : -dt0);     /*  now, regular steps  */
   while ( conds_allow() )  fly( state );

   sprintf(str, "Bounce %d", trials );  BANNER( str );
   record( state );  publish();         /* end of bounced trial */
   G_mode=tmp;  fwd = !fwd;     /*  recover flag & reverse      */
}
```

```
         trials = (fwd ? ++ftrials : ++btrials);        /*  trial over  */
         fflush(fp);  if (hist)  gclose(hp);            /*  flush files */

         /*  recall original values for interface       */
         for ( sptr=storage, dptr=info;  dptr->var!=NULL;
               *((dptr++)->var) = *(sptr++) ) ;
      )  /* (more trials) */
   )  /* (direction) */

   if (ftrials)  fclose( ffp );                 /*  close at run end    */
   if (btrials)  fclose( bfp );
   if (wscreen)  { scrUnwide; wscreen=FALSE; }  /*  unwide screen       */
)  /* (main) */


getdir()        /*  return flight direction     */
{
   printf( "\nDirection to fly (forward, backward, quit):  ");  gets( str );
   if      (str[0]=='f')  { fwd = TRUE;  return( TRUE  ); }
   else if (str[0]=='b')  { fwd = FALSE; return( TRUE  ); }
   else if (str[0]=='q')  {              return( FALSE ); }
   else { printf( "(Need f, b, or q as lst letter)\n" );  return( getdir() ); }
}

publish()       /*  screen output routine       */
{
   /*  print column headers when debugging, screen is full, or done     */
   if ( debug || !(iter % (int)(p_step*(SCR_HEIGHT-4))) || !conds_allow() ) {
      printf( "\n%4s %4s %7s  %3s %4s %5s %5s %5s %5s %5s %5s %6s %6s",
         "Time", " Tau", " N_dot ", "Phi", " V   ", "Gamma", " Psi ",
         "x1(E)", "x2(N)", "x3(H)", "Range", " Lam_1", " Lam_2"); /* 77 */
      if (wide)  printf(" %6s %6s %6s %6s %6s %6s %6s",          /* 49 */
         "Lam_3","Lam_7","Lam_8","Lam_9","Lam10","Lam11","Lam12");/*=126*/
      nl;       /*  (be sure to match column headings with variables!)  */
   }

   /*  screen output:  angles in degrees   */
   printf( "%4.2f %4.2f %7.1f%1s %3.0f %4.0f %5.2f %5.2f",      /*  39+ */
      Time, Tau, N_dot,(squib?"*":" "), deg(phi), V, deg(gamma), deg(psi) );
   printf( " %5.0f %5.0f %5.0f %5.0f %6.0f %6.0f",              /*  38+ */
      x[1], x[2], x[3],  Range,  lam[1], lam[2] );              /*= 77+ */
   if (wide)  printf( " %6.0f %6.0f %6.0f %6.0f %6.0f %6.0f %6.0f",
      lam[3], lam[7],lam[8],lam[9], lam[10],lam[11],lam[12] );  /*  49+ */
   nl;                                                          /*=126+ */

   if (hist) {          /*  file output:  angles in radians     */
      fprintf( vfp, " %.3lf %.3lf %.3lf %.3lf %.3lf %.3lf %.3lf",
         Time,Tau, N_dot,phi, V,gamma,psi );
      fprintf( vfp, " %.3lf %.3lf %.3lf %.3lf\n", x[1],x[2],x[3], Range );
      if (G_mode==2) {  /*  OPTG guidance; show lambda's        */
         fprintf( vfp, " %.3lf %.3lf %.3lf %.3lf %.3lf %.3lf",
            lam[1],lam[2],lam[3], lam[7],lam[8],lam[9] );
         fprintf( vfp, " %.3lf %.3lf %.3lf\n", lam[10],lam[11],lam[12] );
      }  fprintf( vfp, "\n");
   } /* (hist:  use tabs, not spaces if cricket; dbp wants decimal pts) */
```

```
record( state )                 /*  file conditions     */
STATE_VAR state[];              /*  state variables     */
{
 /*  Recording mode:  0:  full states (plus)           */
 /*  (R_mode)         1:  throughout flight            */
 /*                   2:  simple study (a_i vs. b_f)   */
 /*  (overridden by S_mode when "firtree" generating)  */

 int start = !iter;             /*  start of flight     */
 static int Firster=TRUE;       /*  first of 2 trials   */

#define fnl  fprintf(fp, "\n")  /*  file nl macro       */
#define PRINT_St  { LOOP_St( fprintf(fp,"  %g", St(k)); ) fnl; }
/* ------------------------------------------------------------ */

   if (S_mode==5) {     /*  output firtree database     */
      if (trials !=0) { /*  (after initial trial)       */
         if (Firster)   /*  1st flight  */
            if (start)  PRINT_St
            else  fprintf(fp," %g  %g\n", x[1],x[2] );  /*  1st impact  */
         else if (!start)  fprintf(fp," %g  %g\n\n", x[1],x[2] ); /* 2nd */

         if (start)  Firster = !Firster;        /*  toggle      */
      }  /* trials */
   }  /* firtree S_mode */

   else if (fwd && (R_mode==2)) {       /* simple study */
      if (start)  fprintf(fp,"%3d.  %8.2lf  %8.2lf  %8.2lf  ",
                          trials, lam[4], lam[5], lam[6] );
      else        fprintf(fp,"%3d.  %8.11f  %8.11f\n",
                          N, x[1], x[3] );
   }  /*  (decimal points are for the sake of dbprep)   */

   else {       /*  normal R_mode(s)    */
      if (start)  fprintf(fp, " %g  %g  %g  %g  %g  %g  %g  %g  %g\n",
                  W[0],W[1],W[2],W[3],W[4],W[5],W[6],W[7],W[8] );
      PRINT_St
      if (!start) fprintf(fp, " %g   %g   %d   %g   %d   %d\n\n",
                  Time, N_dot, N, Range, Abort, trials );
   }  /*  S_mode  */

   if (!conds_allow()) {        /*  trial done  */
      printf( "\t(Squibs fired: %2d; Distance from origin: %6.1f)\n",
               (fwd ? N : (int)dN_f-N), RSSS(x[1],x[2],x[3]) );
      printf( "\t(Target position:  %8.2f, %8.2f, %8.2f)\n",
               xT[1], xT[2], xT[3] );
      printf( "\t(Targ.-proj. pos:  %8.2f, %8.2f, %8.2f)\n",
               xT[1]-x[1], xT[2]-x[2], xT[3]-x[3] );
   }
   fflush( fp );        /*  close observation   */
}  /*  record  */
```

```
/*  fly.c       */

#include "standard.h"           /*  standard functions, consts  */
 /*  (io & math fns, EPS)       */
#include "integrate.h"          /*  integration structure       */
 /*  (STATE_VAR, N_STATES, FOR3(), St())                        */
#include "system.h"             /*  system-specific constants   */
 /*  (system: S,m;  environment: g,rho)                         */


 /*  Global variables                    */
 extern double W[9];                     /*  guidance parameters */
 extern double dt0, dt, Time, duration; /*  time variables      */
 extern int Abort, debug, fwd;          /*  flags               */
 extern int N, squib;                   /*  #squibs fired, flag */
 extern int G_mode;                     /*  guidance mode       */
 extern int hunting;                    /*  dt iteration (main) */
 extern double Tau;                     /*  T_go estimate (dyn) */

 /*  states                      */
 extern double phi;              /*  rotation angle (rad)        */
 extern double x[4];             /*  position vector (1-3: ENH)  */
 extern double V, gamma, psi;    /*  velocity vector (ft/s, rad) */
 extern double lam[13];          /*  lambdas (1-12; 4-6 const)   */
 extern double xT[4], vT[4];     /*  target position, velocity   */

 /*  state derivatives                       */
 extern double P;                            /*  orientation */
 extern double x_dot[4];                     /*  position    */
 extern double V_dot, gamma_dot, psi_dot;    /*  velocity    */
 extern double lamdot[13];                   /*  (4-6 const) */
 extern double uT[4];            /*  (actual) target accel.      */
 extern double uTm[4];           /*  (measured) target accel.    */


 /*  File variables:  maneuver constants         */
 double Tstar, C0, C2, C3;


conds_allow()                   /*  stop conditions routine     */
{
 int shared = (!Abort && (fabs(dt) >= 0.0005) && (x[3] >= -0.5));

   if (fwd)  return ( shared && (Time<duration) );
   else      return ( shared && (Time>0.0     ) );
}
```

```
fly( state )                        /*  One iteration of the system */
STATE_VAR state[];                  /*  state variables             */
{
 STATE_VAR hypo[N_STATES];          /*  Hypothetical (fwd) state    */

 double T_left;                     /*  bkwd est of time remaining  */
 double val;                        /*  variable to end at zero     */
 int check,passed,converging;       /*  dt iteration case flags     */
 extern int S_last;                 /*  dt iteration var (main)     */
 extern int integ;                  /*  integration (0:rec; 1:Heun) */

 extern int R_mode, p_step,iter;        /*  record mode, step       */
 extern double r_time, random();        /*  random obs time, fn     */
#define R_MAX 1.0                   /*  max sec between recordings  */

#define LIST(p,q) if (debug) {printf(p); LOOP_St(printf("%d:%g  ",k,q(k));) nl;}
/* --------------------------- */

   /*  output when printstep agrees & not aborted or hunting      */
   if (!( (iter % p_step) || Abort || hunting))  publish();
   if ((R_mode==1) && (r_time<=dt0*iter)) /*  time to record      */
      { record( state );  r_time += random(2*dt0, R_MAX); }
   LIST( "states: ", St )  LIST( "dots: ", St_DOT )     /* dbg   */


   check = TRUE;          /*  reduce time_step magnitude near end */
   if      (fwd && (R_mode!=2))          /*  goal: time_to_go==0   */
      { val = T_left = Tau; }
   else if (fwd && (R_mode==2))          /*  goal: x2==xT2         */
      { val = x[2];  T_left = (xT[2]-x[2])/x_dot[2]; }
   else if (!fwd && (gamma > 0.0))       /*  goal: V==V_muzzle     */
      { val = V;     T_left = (3821.5 - V)/V_dot;     }
   else  check = FALSE;             /* (impact half of bkwd flight) */
   /* (note: these equations depend on roughly parabolic paths) */

   if (check) {  /*  look for dt-reducing cases */
      passed = (signum(val) != S_last);          /* goal passed   */
      converging = (fabs(T_left) < fabs(dt));    /*step shrinking*/

      if (hunting || converging || passed) {
         hunting = TRUE;          /*  (dt reduction guaranteed)    */
         if  (converging)  dt = T_left; /*  trust prediction       */
         else if (passed)  dt *= -0.5;  /*  go back half           */
         else              dt *=  0.5;  /*  go forward half        */
      }
   }  S_last = signum(val);      /*  (save for next pass)          */


   /*  Update time stamp and integrate effective Ndot    */
   iter++; Time += dt;  if (squib && !hunting) if (fwd) N++; else N--;
   if (integ==0) {       /*  Integrate states using rectangular method   */
      LOOP_St( St(k) += dt*St_DOT(k); )                      deriv(state);
   }
   else {        /*  Integrate states using Heun's method (1 fwd est.)   */
```

```
      /*  get hypothetical state through rectangular integration        */
      LOOP_St( hypo[k].u = St(k) + dt*St_DOT(k); )          deriv(hypo);
      /*  use average of forward and current derivatives as slope       */
      LOOP_St( St(k) += 0.5*dt*(St_DOT(k) + hypo[k].up); )  deriv(state);
   }
   /* :states & dots up-to-date in both forms after LOOP_St->deriv pairs*/
} /* (fly) */
```

```
finalize()      /*  set final V, lambda values  */
{
 double CD0, CNa, K;            /* Mach "consts"*/
 double qS;                     /* dyn pressure */
 double atmp,btmp,ctmp,dtmp;    /* qS work vars */
 double Qa,Qb,Qc, Mroot, VV;    /* qS work vars */
 int neg;       /* negative discriminant flag   */

#define Vmin 100.0              /*  minimum V_f */
/* --------------------------- */

   /* lambda 5 and 6 are constant (and from interface)  */
   FOR3( lam[k+6] = lam[k+9] = 0.0; )   /*  7->12 are 0 */


   /*  solve for qS_f to get Vf */
   /*  xxx (temporarily use Mach=1.75 values for CNa,K,CD0)     */
   CNa=5.29;  K=1.336;  CD0=0.536;      /* (see tables in dyn)  */

   atmp = (lam[5]*CNa - lam[6]*C2)*C0/Tstar;     btmp = CD0/m;
   ctmp = lam[5]*g*cos(gamma) - W[4]*N*N;        dtmp = g*sin(gamma);
   dbg( "ta: %g  tb: %g  tc: %g  td: %g\n", atmp,btmp,ctmp,dtmp );

   C3 = K*( CNa*CNa + C2*C2 )*C0*C0;    /*  new firing constant */

   Qa =   atmp*btmp;
   Qb =   atmp*dtmp + lam[5]*btmp + C3*ctmp/Tstar;
   Qc = lam[5]*dtmp               + C0*ctmp;
   dbg( "Qa: %g  Qb: %g  Qc: %g\n", Qa,Qb,Qc );

   quad( Qa,Qb,Qc, &qS, &Mroot, &neg ); /*  (use plus root)     */

   VV = 2.0*qS/( rho*S );               /* (qS = 0.5*rho*V*V)   */
   if (neg || (VV < Vmin*Vmin)) {
      Abort = TRUE;
      printf( "--> qS_f problem (V*V = %g); try again.\n\n", VV );
      V = qS = EPS;                     /* (just to finish out iter)    */
   }
   else V = sqrt( VV );


   /*  solve for lambda 4       ( = -(lam5*Q5 + lam6*Q3)/Q0 )   */
   lam[4] = ( qS*C0*( lam[5]*CNa - lam[6]*C2 ) + lam[5]*Tstar )
            /( qS*C3 + Tstar*C0 );
   printf( "Lambda_4f: %g\n", lam[4] );


   /*  now assign lambda's 1-3  */
   lam[1] = -lam[4]/m;
   lam[2] = -lam[6]/m;
   lam[3] = -lam[5]/m;

} /* (finalize) */
```

```
deriv( state )          /*  obtain state derivatives     */
STATE_VAR state[];      /*  state variables              */
{
 extern double N_dot, Nd_thresh;        /*  firing variable, threshold  */
 extern double C0, CNa, K;              /*  Mach-dependent variables    */

 double H3,H5, Q0,Q3,Q5;                /*  work vars   */
 double s_gamma,c_gamma;                /*  work vars   */
 double qS, s_phi,c_phi;                /*  work vars   */
 double CHH,  A0, A1, B;                /*  OPTG vars   */
 double Uc[4], Ndc_psi,Ndc_gam;         /*  baseline    */
 double d_V_dot,d_gamma_dot,d_psi_dot;  /*  delta dots  */
 double L26, L35;                       /*  delta L vars*/

#define Nmax 4000       /*  maximum #squibs (xx no lim) */
/* ------------------- */

   pull_st( state );  dyn();    /*  pre-squib state derivatives */
   /* (xx: could put pull_st in dyn again along w/ integrate.h) */

   s_phi = sin(phi);  s_gamma = sin(gamma);
   c_phi = cos(phi);  c_gamma = cos(gamma);

   H3 = C0*( -CNa*s_phi + C2*c_phi );
   H5 = C0*( -CNa*c_phi - C2*s_phi );   /* (H's include new C0) */

   C3 = K*( CNa*CNa + C2*C2 )*C0*C0;    /*  new firing constant */
                                        /*  (C3 = K*SS(H3,H5))  */
   qS = 0.5*rho*S*V*V;
   Q0 = qS*C3 + Tstar*C0;
   Q3 = qS*H3 - Tstar*s_phi;
   Q5 = qS*H5 - Tstar*c_phi;
   dbg( "Q0: %g  Q3: %g  Q5: %g\n", Q0,Q3,Q5 );


   /*  compute N_dot if guiding, not impacting, and squibs left */
   if ( G_mode && !(hunting || (Tau==0.0)) && (( fwd && (N<Nmax)) ||
                                              (!fwd && (N>0   ))) ) {

      CHH = rho*V*S*( lam[1]*C3 + lam[2]*H3 + lam[3]*H5 );

      if (G_mode==2) {          /*  OPTG guidance               */
         /*  Calculate N_dot:  A2*Nd_dot + A1*N_dot + A0*N = B  */
         /*  (Note:  coefficient signs opposite those of notes) */
         /*  (A2 = -2*W0*Tau^rr, but Nd_dot is n.e. zero)       */
         A1 = 2.0*W[0];         /*  2.0*W[0]*rr*Tau^(rr-1)      */
         A0 = 2.0*W[4];

         B = lamdot[1]*Q0 + lamdot[2]*Q3 + lamdot[3]*Q5
           + V_dot*CHH + P*( lam[2]*Q5 - lam[3]*Q3 );
         /* (...whether we use derivatives with or w/o N_dot)   */

         N_dot = ( B - A0*N )/A1;       /*  or (B/2 - N*W4)/W0  */
         dbg( "A0: %g  A1: %g  B: %g\n", A0,A1,B );
      }

      else {    /*  G_mode==1 --> baseline guidance             */
         /*  acceleration commands zeroing predicted miss*/
         FOR3( Uc[k] = uTm[k] + 2.0*(vT[k]-x_dot[k] + (xT[k]-x[k])/Tau)/Tau;
               dbg( "Uc%d: %g  ", k,Uc[k] ); )  dbg("\n");
```

```
      if (fabs(Q3) < EPS)  Ndc_psi = 0.0;
      else  Ndc_psi = m* (Uc[2]*sin(psi) - Uc[1]*cos(psi))/Q3;
      if (fabs(Q5) < EPS)  Ndc_gam = 0.0;
      else  Ndc_gam = m*((Uc[2]*cos(psi) + Uc[1]*sin(psi))*s_gamma
                        -                  (Uc[3]   +    g)*c_gamma)/Q5;

      N_dot = Ndc_gam*fabs(c_phi) + Ndc_psi*fabs(s_phi);
      dbg( "Ndc_psi: %g  Ndc_gam: %g\n", Ndc_psi, Ndc_gam );
    } /* (G_mode) */

    squib = (N_dot > Nd_thresh);       /*  N_dot large enough? */
  } /* (G_mode, etc.) */


  if (squib) { /*  physical effects of squib (treat N_dot as 1)*/
    d_V_dot     = -Q0/m;               V_dot     += d_V_dot;
    d_gamma_dot = -Q5/(m*V);           gamma_dot += d_gamma_dot;
    d_psi_dot   = -Q3/(m*V*c_gamma);   psi_dot   += d_psi_dot;
    /*  (gamma != k*pi/2)     */

    if (G_mode==2) {                   /*  update lamdots 1-3  */
      L26 = m*lam[2] + lam[6];
      L35 = m*lam[3] + lam[5];         /* (L14 not needed)     */

      lamdot[1] += ( CHH + L35*d_gamma_dot + L26*d_psi_dot*c_gamma )/m;

      lamdot[2] += L26*( V*d_gamma_dot*s_gamma - d_V_dot*c_gamma )
                     /( m*V*c_gamma );

      lamdot[3] += ( -L35*d_V_dot - L26*V*d_psi_dot*s_gamma )/( m*V );
    } /* (G_mode 2) */
  } /* (squib) */


  /*  sequester the state derivatives in the structure */
                                St_DOT( 7) = lamdot[ 1];
  St_DOT( 0) = P;               St_DOT( 8) = lamdot[ 2];
                                St_DOT( 9) = lamdot[ 3];
  St_DOT( 1) = x_dot[1];
  St_DOT( 2) = x_dot[2];        St_DOT(10) = lamdot[ 7];
  St_DOT( 3) = x_dot[3];        St_DOT(11) = lamdot[ 8];
                                St_DOT(12) = lamdot[ 9];
  St_DOT( 4) =     V_dot;
  St_DOT( 5) = gamma_dot;       St_DOT(13) = lamdot[10];
  St_DOT( 6) =   psi_dot;       St_DOT(14) = lamdot[11];
                                St_DOT(15) = lamdot[12];

  St_DOT(16) = vT[1];  St_DOT(19) = uT[1];
  St_DOT(17) = vT[2];  St_DOT(20) = uT[2];
  St_DOT(18) = vT[3];  St_DOT(21) = uT[3];
} /* (deriv) */
```

```
set_st( state )          /*  set states in integration structure for dyn */
STATE_VAR state[];       /*  input:  state variables      */
{
   St(0)= phi;
   St(1)=x[1];  St(4)=    V;  St(7)=lam[1];  St(10)=lam[7];  St(13)=lam[10];
   St(2)=x[2];  St(5)=gamma;  St(8)=lam[2];  St(11)=lam[8];  St(14)=lam[11];
   St(3)=x[3];  St(6)=  psi;  St(9)=lam[3];  St(12)=lam[9];  St(15)=lam[12];

   St(16)=xT[1];  St(19)=vT[1];
   St(17)=xT[2];  St(20)=vT[2];
   St(18)=xT[3];  St(21)=vT[3];
}


pull_st( state )         /*  pull states from integration structure      */
STATE_VAR state[];       /*  input:  state variables                     */
{
   static double circle=2.0*PI;
   St(0) -= circle*(int)(St(0)/circle);          /*  phi wrap-around     */
   if (St(0)<0.0)  St(0) += circle;              /*  positive version    */

   phi =St(0);
   x[1]=St(1);  V    =St(4);  lam[1]=St(7);  lam[7]=St(10);  lam[10]=St(13);
   x[2]=St(2);  gamma=St(5);  lam[2]=St(8);  lam[8]=St(11);  lam[11]=St(14);
   x[3]=St(3);  psi  =St(6);  lam[3]=St(9);  lam[9]=St(12);  lam[12]=St(15);

   xT[1]=St(16);  vT[1]=St(19);
   xT[2]=St(17);  vT[2]=St(20);
   xT[3]=St(18);  vT[3]=St(21);
}


tconst()         /*  set P- and dt-dependent constants   */
{                /*  (called in main prior to first dyn) */
   Tstar = 0.75/dt0;             /*  I_s/dt_decision     */
   C0 = 6.5205e-6*Tstar/dt0;     /* -new squib constant  */
   C2 = P*8.1938e-4;             /*  P*0.5*CYp*d_ref/Vav */
}
```

```
/*  advise.c    */

#include "standard.h"
#include "interface1.h"          /*  interface1 structure DESCRIPTION    */
#include <GetfilePkg.h>          /*  GETFILE, FP, GFILE package          */


 /*  Global variables from main */
 extern double W[9];             /*  guidance parameters */
 extern double duration;         /*  maximum flight time */
 extern double score;            /*  grope metric        */
 /*  extern double N_dot;         *  control variable    */

 /*  state variables                  */
 extern double phi;                    /*  rotation angle (rad)          */
 extern double x[4];                   /*  position vector (1-3; ENH)    */
 extern double V, gamma, psi;          /*  velocity vector (ft/s, rad)   */
 extern double lam[13];                /*  lambdas (1-12; 4-6 [illegible]  */
 extern double xT[4], vT[4];           /*  target position, velocity     */

 extern int S_mode;      /*  search mode:                  */
#define GRID    1        /*  explore regular grid          */
#define GROPE   2        /*  use grope for search          */
#define GAUSS   3        /*  explore gaussian space        */
#define UNIF    4        /*  explore unif. random space    */
#define FIRTREE 5        /*  generate fwd-branching db     */
#define loopK(h)  {int j;  for (j=0; j<numK; j++) {h} }


 /*  Global variable for record */
 /* double Tgo0;                  *  firtree base time to go     */


more_trials( fwd, trials, info )/*  routine governing trials     */
 int fwd, trials;                /*  direction flag, #trials     */
 struct DESCRIPTION *info;       /*  ptr to interface structure  */
{
#include "bounds.h"              /*  Min_, Max_, Inc_, num_ (F,B)*/

 double K[max(numF,numB)];       /*  parameter values    */
 int numK;                       /*  index, #parameters  */
 int advise(), interface();      /*  routines            */
 int GOING;                      /*  returned flag       */
 static int First_auto=TRUE;     /*  auto. mode flag     */
 /* GFILE *gp; static FILE *firp;*  firtree file ptr    */
 /* static int count=0;           *  branch counter      */
 /* double tmp;                   *  read place-holder   */
 /* define DELTA_G  50            *  G4,G5 change        */
 /* -------------------------- */

   if (!S_mode) {        /*  use interface       */
      char *title="129-00 TPBV Guidance Project";
      GOING = interface( title, info, (trials ? NORESET : RESET) );
      /* ('quit' exits) */      /* (reset if no previous trials)*/
   }
```

```
   /* xx (firtree mode currently disabled)
   else if (S_mode==FIRTREE) {    *  adjust params through database      *
      if (First_auto) {           *  open reference data file    *
         First_auto = FALSE;
         GETFILE( gp, "Base data filename: ", "base.db", "r");
         firp = FP( gp );         *  file pointer        *
      }

      if ( (count%2)==0 ) {       *  first trial of new flight   *
         printf("Reading conditions for reference observation %d.\n", count/2);
          *  (reference assumed to impact at (0,0))     *
         GOING  =( 3==fscanf(firp,"%lf %lf %lf", &G4,&G5,&Tgo0) );
         GOING &=(10==fscanf(firp,"%lf %lf %lf %lf %lf %lf %lf %lf %lf %lf",
                  &lam[6],&lam[1],&lam[2],&lam[3],
                  &V,&gamma,&psi, &x[1],&x[2],&x[3]) );

         if (GOING)  G4 += DELTA_G;
         else  printf( "-->  Reading Done  <--\n" );
      }
      else {              *  second trial        *
         GOING = TRUE;
         G4 -= DELTA_G;           *  reset G4    *
         G5 += DELTA_G;           *  change G5   *
      }

      printf( "Try G4: %g and G5: %g.\n", G4, G5 );
      count++;
   }
   (firtree) xx */


   else {       /*  adjust parameters automatically     */
      numK = (fwd ? numF : numB);

      if (First_auto) { /*  allow user to verify limits */
         First_auto = FALSE;
         printf( "Minimums, maximums (and increments) are:\n" );  loopK(
            if (fwd) printf( "%d: %g\t%g\t(%g)\n", j, MinF[j],MaxF[j],IncF[j] );
            else     printf( "%d: %g\t%g\t(%g)\n", j, MinB[j],MaxB[j],IncB[j] );
         )
      }

      /* (Note:  vars in both directions must match those in bounds.h)  */
      if (fwd) {        /* Target East, Height  */
         K[0]=lam[4];  K[1]=lam[5];  K[2]=lam[6];
         GOING = advise( numK, K, MinF, MaxF, IncF );
         lam[4]=K[0];  lam[5]=K[1];  lam[6]=K[2];
      }
      else {
         K[0]=lam[5];  K[1]=lam[6];  K[2]=W[4];
         GOING = advise( numK, K, MinB, MaxB, IncB );
         lam[5]=K[0];  lam[6]=K[1];  W[4]=K[2];
      }
   } /* (S_mode) */

   return( GOING );
} /* (more_trials) */
```

```
advise( numK, K, Min, Max, Inc )
/*  advise more_trials about parameter values   */
 int numK;              /*  #parameters          */
 double K[];            /*  array of parameters */
 double Min[], Max[];   /*  bounds               */
 double Inc[];          /*  increments (if GRID)*/
{
 static int GOING=TRUE; /*  FALSE when over      */
 static int a_count=0;  /*  calls counter        */
#define Max_Iter  1000  /*  maximum a_count      */
 double half;           /*  used for mean, sigma*/
 int cascade(), grope();  double gauss(), random();
 /* ------------------- */
   a_count++;

   /*  distribute samples ...   */
   if      (S_mode==GRID ) {           /*  ... regularly       */
      if (a_count==1)  loopK( K[j] = Min[j]; )
      else GOING = cascade( 0,numK-1, K, Min,Max,Inc );
   }
   else if (S_mode==GROPE) {           /*  ... with a purpose  */
      GOING = !grope( score, numK, K, Min, Max );
      /*  (grope counts iterations on its own)  */
   }
   else if (S_mode==GAUSS) {           /*  ... gaussianly      */
      GOING = (a_count <= Max_Iter);
      if (GOING)  loopK( half = (Max[j] - Min[j])/2.0;
                         do  K[j] = (Min[j] + half) + gauss( half/2.0 );
                         while ( (K[j] < Min[j]) || (K[j] > Max[j]) );  )
   }
   else if (S_mode==UNIF ) {           /*  ... uniformly       */
      GOING = (a_count <= Max_Iter);
      if (GOING)  loopK( K[j] = random( Min[j], Max[j] ); )
   }
   else {
      printf( "--> Unknown search mode: %d (%g) <--\n", S_mode, S_mode );
      GOING = FALSE;
   }

   if (GOING) {
      printf( "--> Try values:  ");  loopK( printf("%g  ",K[j]); )  nl;
   }
   else {
      S_mode = FALSE;           /*  put user in interface       */
      printf( "\n--> (Restart if wishing to search further) <--\n" );
   }
   return( GOING );
) /* (advise) */
```

```
cascade( j,jm, K, Min,Max,Inc ) /*  roll down "inverted for loops"       */
 int j, jm;                      /*  current, maximum index               */
 double K[], Min[],Max[],Inc[];
{
   K[j] += Inc[j];
   if ((j < jm) && (K[j] > Max[j])) {
      K[j] = Min[j];
      cascade( j+1,jm, K, Min,Max,Inc );
   }
   return( K[jm] <= Max[jm] );  /*  true while continuing       */
}
```

```
/*  dyn.c       */

#include "standard.h"                    /*  standard functions, consts  */
 /*  (stdio.h, math fns, max(), dbg)     */
#include "system.h"                      /*  system-specific constants   */
 /*  (system: S,m;  environment: g,rho) */
 double CD0, K, CNa;                     /*  Mach-dependent system vars  */
                                         /*  calculated from tables here */

 /*  Global variables for others:  state derivatives     */
 /*  (note: orientation (phi) & P (phi_dot) not needed) */
 double x_dot[4];                        /*  position     */
 double V_dot, gamma_dot, psi_dot;       /*  velocity     */
 double lamdot[13];                      /*  (4-6 const) */
 double uTm[4];                          /*  measured uT */


 /*  Global state variables from main    */
 extern double x[4];                     /*  position vector (ft) (1-3)  */
 extern double V, gamma, psi;            /*  velocity vector (ft/s, rad) */
 extern double lam[13];                  /*  lambdas (1-12; 4-6 const)   */
 extern double xT[4], vT[4];             /*  target position, velocity   */

 extern double uT[4];            /*  constant target acc */
 extern double Umean, Usig;      /*  noise mean, sigma   */
 extern double gauss();          /*  rng for acc noise   */


 dyn()                   /*  given states, calculate state derivatives   */
{
          /* ---------------------------------- */
         /*      Generalized 3-DOF Dynamics      */
        /*    Copyright 1987 Barron Assoc., Inc.  */
         /*           John F. Elder IV           */
          /* ---------------------------------- */

 /*  Global variables from main          */
 extern double W[9];                     /*  guidance parameters (0-5)   */
 extern double SW[4];                    /*  guidance parameters (6-8)   */
 extern double Tau, Range;               /*  Est time, dist to target    */
 extern int debug, G_mode;               /*  debugging, guidance flags   */


 /*  internal variables and constants                    */
 double v[4], Uc[4];     /*  projectile vel, command acc.*/
 double d[4], e[4];      /*  position, vel differences   */
 double det[4];          /*  Taylor series diff. w/o U's */
 double s_gamma,s_psi, Lsc,Ucsc;         /*  work vars   */
 double c_gamma,c_psi, Lcs,Uccs;         /*  work vars   */
 double qS, TT, L26,L35, tmp,tmp2;       /*  work vars   */


 double Mach;                    /*  Mach # wrt V_sound at alt 0 */
 int ind; double prop;           /*  interpolation variables     */
 static double Mbound[11]={ 0.0,   0.5,   0.9,   1.1,   1.2,   1.5,
                            2.0,   2.5,   3.0,   3.5,   4.0   };
 static double CD0val[11]={ 0.450, 0.467, 0.514, 0.582, 0.631, 0.572,
                            0.500, 0.444, 0.394, 0.336, 0.308 };
```

```
 static double   Kval[11]={ 1.300, 1.335, 1.362, 1.569, 1.768, 1.562,
                            1.110, 0.940, 0.942, 0.858, 0.812 };
 static double CNaval[11]={ 5.03,  5.03,  5.03,  5.03,  5.03,  5.03,
                            5.55,  5.61,  5.55,  5.35,  5.01  };
 /* (values at Mach=0 added by hand; could instead use Glauert approx)  */
 /* Glauert (below Mach 1) or Prandtl (above Mach 1) approximations:        */
 /* if      (Mach<Mbound[ 0]) CD0_M = Cdrag0/sqrt(1.0 - Mach*Mach);         */
 /* else if (Mach>Mbound[10]) CD0_M = Cdrag0 + 0.1415/sqrt(Mach*Mach - 1.0); */

 int k;          /*  macro loop index    */
#define FOR3(h)  for (k=1; k<=3; k++) {h}
/* ------------------------------------ */

   /*  adjust drag coefficient according to Mach number profile */
   Mach = V/1116.89;     /*  (using sound velocity @ 0 altitude) */
                         /*  (max Mach of 4.0 -> max V=4457.56)  */
   /*  find lower bound index and proportion of overflow         */
   for( ind=0; Mach>Mbound[ind+1]; ind++);
   prop = ( Mach-Mbound[ind] )/( Mbound[ind+1] - Mbound[ind] );

   CD0  =   CD0val[ind] + prop*( CD0val[ind+1] - CD0val[ind] );
   CNa  =   CNaval[ind] + prop*( CNaval[ind+1] - CNaval[ind] );
   K    =     Kval[ind] + prop*(   Kval[ind+1] -   Kval[ind] );
   dbg( "Mach: %g;  CD0: %g;  CNa: %g;  K: %g\n", Mach, CD0, CNa, K );


   qS = 0.5*rho*V*V*S;
   s_gamma = sin(gamma);   s_psi = sin(psi);
   c_gamma = cos(gamma);   c_psi = cos(psi);


   /*  Geographic rates */
   v[1] = x_dot[1] = V*c_gamma*s_psi;   /*  "East"      */
   v[2] = x_dot[2] = V*c_gamma*c_psi;   /*  "North"     */
   v[3] = x_dot[3] = V*s_gamma;         /*   up         */


   /*  Acceleration vector (sans squib effects) */
   V_dot     = -g*s_gamma - qS*CD0/m;   /* ft/ss*/
   gamma_dot = -g*c_gamma/V;            /* rad/s*/
   psi_dot   =  0.0;                    /* rad/s*/


   /*  simple Tau estimation = Range/(V along Range)     */
   tmp = tmp2 = 0.0;     /*  use pos and vel differences */
   FOR3( d[k] = xT[k]-x[k];     tmp  += d[k]*d[k];
         e[k] = vT[k]-v[k];     tmp2 -= d[k]*e[k]; )
   Range = sqrt( tmp );

   if (Range==0.0)  Tau = 0.0;  /*  impact      */
   else  Tau = (Range/( V - 0.001695*sqrt(V)*Range ))*signum(tmp2);
   /*  (const ideal for 10K ft, 3.605 sec flight with 32 squibs)       */
   /*  (signum is for iteration purposes)      */


   /*  measure (with noise) target acceleration */
   FOR3( uTm[k] = uT[k] + Umean + gauss( Usig ); )
```

```
/*  Adjoint guidance equations        */
if (G_mode != 2)                      /*  baseline or no guidance     */

   FOR3( lamdot[k]=lamdot[k+6]=lamdot[k+9] = 0.0; )  /*  (1-3, 7-12) */
   /*  (xx: could do this just at the beginning of each trial)        */

else {                /*  TPBV guidance; OPTG (sans squib effects)    */

   Lcs = lam[4]*c_gamma - lam[5]*s_gamma;
   Lsc = lam[4]*s_gamma + lam[5]*c_gamma;

   /*  calculate commanded accelerations     */
   if (Tau==0.0) {   /*  kinematic eqns      */
      tmp  = V_dot*c_gamma - V*gamma_dot*s_gamma;
      tmp2 =                 V*  psi_dot*c_gamma;
      Uc[1] = tmp*s_psi + tmp2*c_psi;
      Uc[2] = tmp*c_psi - tmp2*s_psi;
      Uc[3] = V_dot*s_gamma + V*gamma_dot*c_gamma;
   }
   else {            /*  variational eqns    */
      TT = Tau*Tau/2.0;
      FOR3( det[k] = d[k] + Tau*e[k] + TT*uTm[k]; )  /* measured uT  */
      /* (TT, det required only when Tau!=0) */

      Uc[1] = Lcs*s_psi + lam[6]*c_psi;      /*  (partial)   */
      Uc[2] = Lcs*c_psi - lam[6]*s_psi;
      Uc[3] = Lsc;
      tmp = TT*W[5]*Tau;     /* 0.5*W[5]*Tau^(4-ss)  */
      FOR3( Uc[k] += W[k]*v[k] + det[k]*W[5]*Tau - lam[k+9];
            Uc[k] /= tmp; )
   }
   FOR3( dbg( "Uc%d: %g  ", k,Uc[k] ); )  dbg("\n");

   Uccs = Uc[1]*c_psi - Uc[2]*s_psi;
   Ucsc = Uc[1]*s_psi + Uc[2]*c_psi;

   L26 = m*lam[2] + lam[6];
   L35 = m*lam[3] + lam[5];  /* (L14 not needed)     */

   lamdot[1] = ( rho*V*S*CD0*lam[1] + L35*gamma_dot )/m;

   lamdot[2] = ( L26*( V*gamma_dot*s_gamma - V_dot*c_gamma )
               - Lcs*Uccs + lam[6]*Ucsc )/( m*V*c_gamma );

   lamdot[3] = ( m*g*( lam[1]*c_gamma - lam[3]*s_gamma )
               - L35*V_dot + Lsc*Ucsc - Lcs*Uc[3] )/( m*V );

   if (Tau==0.0)  FOR3( lamdot[k+6] = 0.0; )
   else {  tmp = 2.0*W[5]/Tau;       /*  (2.0*W[5]*Tau^-ss)  */
           FOR3( lamdot[k+6] = tmp*( det[k] - TT*Uc[k] ); )
   }
   FOR3( lamdot[k+9] = W[k]*Uc[k] + Tau*lamdot[k+6] - lam[k+6] + SW[k]; )

 } /* (G_mode) */
}  /*  dyn  */
```

```
/*  quad.c      */

#include "standard.h"           /*  standard fns, consts*/


 /*  Global variables from main */
 extern int debug;              /*  debug flag          */
 extern int Abort;              /*  fatal error flag    */


quad( A,B,C, proot,mroot, neg ) /*  solve quadradic     */
double A,B,C;                   /*  input:  Axx+Bx+C=0  */
double *proot,*mroot;           /*  output:  +,- roots  */
int *neg;                       /*  output:  error flag */
{
 double tmp, discrim, RR;       /*  work vars           */
/* --------------------------- */

   *neg = FALSE;        /*  until proven otherwise      */

   if ( A==0.0 )        /*  solution can be linear      */
      if ( B==0.0 )  Abort = TRUE;      /* (stop trial) */
      else *proot = *mroot = -C/B;

   else  {              /*  solution can be quadratic   */
      RR = -B/(2.0*A);  /*  real part (if discrim neg)  */
      discrim = RR*RR - C/A;
      if (discrim < 0.0)  {
         *neg = TRUE;   *proot = *mroot = RR;   /*  use real part only  */
         dbg( "\n-->  Warning:  Negative discriminant: %g  <--\n", discrim );
         dbg( "(Components: a: %g; b: %g; c: %g)\n", A,B,C );
         dbg( "(linear (-c/b): %g; using real (-b/2a):  %g)\n", -C/B, RR);
      }
      else  {   /*  real roots  */
         tmp = sqrt( discrim );         *proot = RR + tmp;
                                        *mroot = RR - tmp;
      } /* (discrim) */
   } /* (A) */
   dbg( "plus root:%g, minus root:%g\n", *proot, *mroot );
}  /* quad */
```

```
/*  rng.c       */

double  gauss( std )    /*  pseudo-gaussian pseudo-random number gen'r  */
                        /*  Barron Associates, Inc.     1/9/87          */
double  std;            /*  desired gaussian standard deviation         */
#define NUMRNDS 5       /*  #uniform random numbers to sum to approx    */
{
  extern double sqrt();
  double random(), stduse, sum;  register int i;

    stduse = std * sqrt(3.0/NUMRNDS);

    for (sum=0.0, i=0; i < NUMRNDS; i++)
        sum += random(-stduse, stduse);

    return( sum );
} /* (gauss) */


double random( lo, hi ) /*  uniform pseudo-random number generator      */
double lo, hi;          /*  limits on returned number (order-invariant) */
{
  extern int srand(), rand();
  double tmp, prop;
  static int first_time=1;

    if (first_time) { srand(12347); first_time=0; }     /*  (seed)      */
    if (lo > hi)    { tmp=lo;  lo=hi;  hi=tmp; }        /*  (swap)      */

    do prop =  ((double) rand()) / 32767.0;     /* rand range: 0-2^15-1 */
    while ( (prop < 0.0) || (prop > 1.0) );     /* xx: shouldn't be nec.*/

    return( lo + prop*(hi-lo) );        /*  Distribute between limits   */
}
```

```
/*  standard.h  */

#include <stdio.h>                          /*  standard i/o functions      */
/*  (printf(), fprintf(), gets(), fflush(), strcpy(), strcat())             */
extern double sin(), cos(), fabs(), sqrt();     /*  math.h functions        */
/* (also:  tan(), asin(),acos(),atan(), exp(),log10() pow(), fmod())        */

#define PI  3.14159265359
#define DEGtoRAD  (PI/180.)                 /*  degree to radians converter */
#define rad(h)  DEGtoRAD*(h)                /*  convert to radians          */
#define deg(h)  (h)/DEGtoRAD                /*  convert to degrees          */

#define min(a,b) ( ((a)<(b))?(a):(b) )
#define max(a,b) ( ((a)>(b))?(a):(b) )
#define nl  printf("\n")                    /*  newline     */
#define signum(x) (((x)<0.0) ? -1: 1)       /*  signum fn with 0 positive   */
#define EPS  0.00001                        /*  epsilon ("small enough")    */
/*  define epsig(x) ((fabs(x)<EPS) ? 0 : signum(x))   *signum w/ 0 band */

#define limit(h,lo,hi)  h = (h<(lo)?(lo):( h>(hi)?(hi):h ))  /* bi-limit*/
#define RSS(a,b)    sqrt((a)*(a)+(b)*(b))       /*  root sum squared    */
#define RSSS(a,b,c) sqrt((a)*(a)+(b)*(b)+(c)*(c))       /*  RSS w/ 3    */

#include <environ.h>                        /*  machine-dependent params    */
#define SCR_HEIGHT 24
#define SCR_WIDTH  80
#include <dbg_off.h>                        /*  debug macros (incl. dbg)    */
```

```
/*  integrate.h */

typedef struct  {               /*  a state variable            */
        double  u, up;          /*  current u, u_dot vector     */
}  STATE_VAR;

#define N_STATES 22             /*  number of system states     */

/*  aliases for states and derivatives (use structure "state")  */
#define St(h)       state[h].u
#define St_DOT(h)   state[h].up

 int k;                 /*  macro loop index    */
#define LOOP_St(h)  for (k=0; k<N_STATES; k++) {h}
#define FOR3(h)     for (k=1; k<=3;       k++) {h}
```

```
/***  system.h for 129 Tank projectile  ***/

/*  system constants     */
#define m       0.966     /*  mass (slugs) (31.1 lbs)      */
#define S       0.12151   /*  reference area (ft^2)        */
/*  (see also Mach-related CD0, K, CNa tables in dyn)    */

/*  environment:         */
#define g  32.199         /*  gravity                */
#define rho 0.00237692    /*  atmospheric density */
```

```
/*  init.h:  interface variables          */
/*  (common variables in "fandb.h")       */

/*  Backward flight                       */
 static struct DESCRIPTION backinfo[] = {
#include "fandb.h"
   "N",          &dN_f,       200.,  "Final #squibs",
   "",            NULL,         0.,  "(end_of_structure pointer)"
 } ;

/*  Forward flight                        */
 static struct DESCRIPTION fwdinfo[] = {
#include "fandb.h"
   "lam4",      &lam[4],    200.,  "V_dot lambda (const)",

   "phi",       &phi,         0.,  "body roll attitude (deg)",
   "V",         &V,        3821.5, "velocity magnitude (ft/sec)",
   "psi",       &psi,         0.,  "velocity vector heading (deg)",

   "lam1",      &lam[1],    100.,  "V_dot lambda",
   "lam2",      &lam[2],    100.,  "psi_dot lambda",
   "lam3",      &lam[3],   -100.,  "gamma_dot lambda",
   "lam7",      &lam[7],      0.,  "x1_dot lambda",
   "lam8",      &lam[8],      0.,  "x2_dot lambda",
   "lam9",      &lam[9],      0.,  "x3_dot lambda",
   "lam10",     &lam[10],     0.,  "v1_dot lambda",
   "lam11",     &lam[11],     0.,  "v2_dot lambda",
   "lam12",     &lam[12],     0.,  "v3_dot lambda",

   "W6",        &W[6],        0.,  "downrange weight",
   "W7",        &W[7],        0.,  "crossrange weight",
   "W8",        &W[8],        0.,  "altitude weight",

   "",          NULL,         0.,  "(end_of_structure pointer)"
 } ;

/*   "use_APN",  &duse_APN,   0.,  "Use APN(s) (0:no; 1:yes)", */
/*  (xx make W6,7,8 bi-directional)     */
```

```
/*  fandb.h     */

 "duration",  &duration,  10., "maximum duration of flight (sec)",
 "dt",        &dt0,       .05, "timestep magnitude",
 "print_int", &print_int, .05, "printing interval (sec)",
 "integ",     &dinteg,    0.,  "integration method (0:rectangular; 1:Heun)",
 "wide",      &dwide,     0.,  "wide screen flag (0: 80; 1: 132 columns)",
 "debug",     &ddebug,    0.,  "debug flag (0:normal; 1:debug)",
 "hist",      &dhist,     0.,  "variable history flag (0:none; 1:record)",

 "guide",     &dG_mode,   2.,  "guidance (0:ballistic; 1:baseline; 2:OPTG)",
 "bounce",    &dbounce,  -1.,  "(neg:no bounce; 0-2:bounce with that guidance)",
 "thresh",    &Nd_thresh, 0.,  "N_dot threshold above which firing occurs",
 "record",    &dR_mode,   0.,  "recording (0:end states; 1:during; 2:study)",
 "search",    &dS_mode,   0.,  "0:none; 1:grid; 2:grope; 3:gauss; 4:unif; 5:fir",

 "P",         &P,         21., "roll rate (Hz)",
 "gamma",     &gamma,      2., "velocity vector inclination (deg)",

 "xT1",       &xT[1],    100., "Target eastward position (ft)",
 "xT2",       &xT[2],  10000., "Target northward position (ft)",
 "xT3",       &xT[3],    200., "Target altitude (ft)",
 "vT1",       &vT[1],      0., "Target eastward velocity (ft/sec)",
 "vT2",       &vT[2],      0., "Target northward velocity (ft/sec)",
 "vT3",       &vT[3],      0., "Target rate of altitude increase (ft/sec)",
 "uT1",       &uT[1],      0., "Target eastward acceleration (ft/sec^2)",
 "uT2",       &uT[2],      0., "Target northward acceleration (ft/sec^2)",
 "uT3",       &uT[3],      0., "Target altitude acceleration (ft/sec^2)",
 "Umean",     &Umean,      0., "mean target acceleration noise (f/sec^2)",
 "Usig",      &Usig,       0., "target acceleration noise sigma (f/sec^2)",

 "W0",        &W[0],     400., "N_dot scaling weight",
 "W1",        &W[1],       0., "Eastward kinetic energy use weight",
 "W2",        &W[2],       0., "Northward kinetic energy use weight",
 "W3",        &W[3],       0., "Altitude kinetic energy use weight",
 "W4",        &W[4],     100., "Squib use (N) weight",
 "W5",        &W[5],   0.001, "Predictive error penalty weight",

 "lam6",      &lam[6], 10000., "psi_dot lambda (const)",
 "lam5",      &lam[5], 10000., "gamma_dot lambda (const)",
```

```
/*  file:  bounds.h     */

#define numF 3  /*  vary fwd:    _lam4_     _lam5_     _lam6_       */
 static double MinF[numF] = {  -150000., -150000., -150000.      };
 static double MaxF[numF] = {   200000.,  100000.,  150000.      };
 static double IncF[numF] = {    50000.,   50000.,   50000.      };

#define numB 3  /* vary bkwd:  lam_5    lam_6     _W4_    */
 static double MinB[numB] = { -10000., -10000.,  100. };
 static double MaxB[numB] = {  10000.,  10000.,  600. };
 static double IncB[numB] = {   4000.,   4000.,  100. };
```

```
#---------------------- 129 3dof makefile --------------------------+

# make - compile the program
# make lint - lint in the background
# make go - run the program
# make all - lint in the background and then run
#            (note:  gives you too many prompts for unknown reason)
#-------------------------------------------------------------------
#makefile template for Barron Associates, Inc.  7/30/87 (dwa, ph)

#name of final application
PROG = 3dof.out

#name of each object file
OBJ = main.o fly.o dyn.o advise.o
OBJ2 = rng.o quad.o grope.o step.o matrix.o
#also  alt.o

#list of standard libraries to use
STDLIBS = -lc -lm

#name of BAI library(ies) (now in standard place - 8/3/87/ph)
#BAILIBS = -p /usr/bai/lib/libBAI.a
BAILIBS = -lBAI

#location of BAI include files
INC = /usr/include

#----------------------------- commands ------------------------------------
#the default (just type 'make')
$(PROG): $(OBJ) $(OBJ2) /usr/lib/libBAI.a
        cc -o $@ $(OBJ) $(OBJ2) $(STDLIBS) $(BAILIBS)

# 'make go' will run.
go: $(PROG)
        nice $(PROG)

# 'make lint' will strongly lint all in background
lint:
        back lint -abchx *.c

# 'make all' will lint then run
all: lint go

#--------------------------- dependencies -------------------------------

main.o:     integrate.h  standard.h    init.h  fandb.h  $(INC)/interfacel.h
fly.o:      integrate.h  standard.h  system.h
advise.o:                standard.h            bounds.h  $(INC)/interfacel.h
dyn.o:                   standard.h  system.h
quad.o:                  standard.h
grope.o, step.o:         standard.h   grope.h
matrix.o:                             grope.h

# alt.o:                 standard.h
# apn0.o, apnI.o:                       apn.h
```

END

DATE

FILMED

6-1988

DTIC